VPFLOW: VALUE-PREDICTIVE PLANNING WITH IMPERFECT WORLD MODELS FOR OFFLINE GOAL REACHING

Anonymous authors
Paper under double-blind review

ABSTRACT

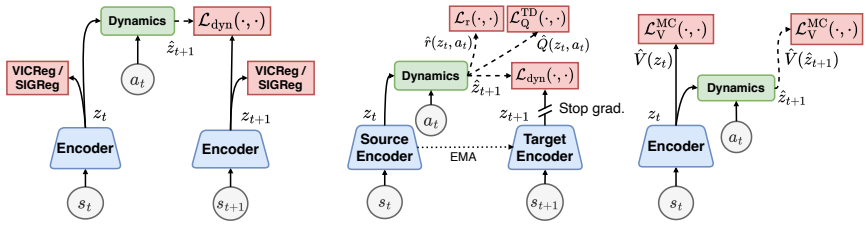
Offline goal-conditioned reinforcement learning turns unlabeled trajectories into reward-free skill acquisition: relabel future observations as goals, and one dataset supervises many tasks. Model-based planning could add test-time compute on top, but existing planners share an assumption: the world model must predict the next state, in pixels or in latents, faithfully enough to plan with. We challenge this assumption. Offline goal reaching admits a closed-form Monte Carlo return under sparse rewards, so value learning becomes regression instead of temporal-difference bootstrapping; we use this return as the sole supervision for the latent dynamics, training imagined latents to predict goal-reaching value rather than the next state, which prevents representation collapse without stop-gradients or target encoders. We present VPFLOW, which pairs this deliberately imperfect world model with a flow policy and steers it at test time in the policy's noise space, avoiding the out-of-distribution action queries of action-space planning. On 30 OGBench datasets with state or pixel inputs, VPFLOW raises mean success from 32.9% to 63.2% (state) and from 27.5% to 37.6% (pixel) over the strongest baselines, even though its state-prediction error is orders of magnitude larger and its imagined latents stray far from the manifold of encoded states; reducing that error lowers success. Planning needs value-predictive representations, not faithful simulation.

1 INTRODUCTION

Offline reinforcement learning seeks to learn reusable behaviors purely from previously collected trajectories with no additional environment interaction (Levine et al., 2020). In its goal-conditioned setting, an agent is tasked with reaching a desired goal at test time without optimizing a hand-designed reward (Kaelbling, 1993; Schaul et al., 2015; Park et al., 2025a). This makes offline goal reaching especially attractive for reward-free skill acquisition: relabeling future observations as goals turns one unlabeled dataset into supervision for many tasks (Andrychowicz et al., 2017). Pairing it with model-based planning is a natural next step: pre-train on offline data once, then adapt to new goals in zero-shot by searching over imagined futures at test time. The combination is promising, yet comparatively unexplored.

Strong offline goal-conditioned baselines adapted from implicit Q-learning (Kostrikov et al., 2022), quasi-metric representations (Wang et al., 2023; Myers et al., 2025), and contrastive learning (Eysenbach et al., 2022) learn a policy or value function from relabeled trajectories and act greedily at test time. They are robust and efficient, but fundamentally policy-extractive: once training ends, additional test-time compute cannot improve reasoning on a hard instance. Model-based planning offers the complementary advantage. From Dyna (Sutton, 1991) to PlaNet (Hafner et al., 2019), Dreamer (Hafner et al., 2020), MuZero (Schrittwieser et al., 2020), and TD-MPC (Hansen et al., 2022; 2024), learned models convert extra inference-time compute into better decisions by searching over imagined futures. But these methods are built around a common assumption that has migrated from pixels to latents largely unquestioned: the world model must predict the next state, or the next latent state, faithfully enough before planning becomes reliable (Hafner et al., 2019; Schwarzer et al., 2021; Sobal et al., 2025; Maes et al., 2026).

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.



(a) Joint Embedding Predictive Architecture (JEPA) (b) TD-MPC (c) VPFlow (Ours)

Input	$\{s_t, a_t, s_{t+1}\}$	$\{(s_k, a_k, r_k, s_{k+1})\}_{k=t}^{t+H-1}$	$\{s_t, a_t, a_{t+1}, \ldots, a_{t+H-1}\}$
Encoder Type	Joint	Source, Target	Single
Value Learning	None	Temporal Difference (1-step TD)	Monte Carlo Return Regression
Latent Regularization	Explicit (e.g., VICReg, SIGReg)	Explicit (target network, stop gradient, EMA)	Implicit (value regression)
Reward-free	✔	✘	✔

Figure 1: **State-predictive and value-predictive action-conditioned latent world models.** *Left*: JEPA uses a joint encoder with explicit regularization to align predicted and future latent representations. *Middle*: TD-MPC uses source and target encoders with stop gradient and EMA heuristics. *Right*: VPFLOW uses a single encoder and trains the latent dynamics via value prediction: predicted latent states seek accurate goal-reaching value estimates, not future states. Closed-form Monte Carlo targets naturally provide temporal signal as implicit regularization.

Our starting point is that this assumption is both too strong and misdirected. What a planner needs is not whether the model can reproduce the next frame, but whether it preserves the ordering of candidate futures by how much they improve goal-reaching value. This view has a lineage. Value prediction networks learn a model only insofar as it reproduces Bellman backups (Oh et al., 2017); universal value functions and successor representations factor value into a goal-agnostic prediction of future occupancy and a goal-dependent readout (Schaul et al., 2015; Dayan, 1993; Barreto et al., 2017); and the value-equivalence principle formalizes when a model that is wrong as a simulator is still right for control (Grimm et al., 2020). We take the step further: a predicted latent only has to land on the right level set of the value field, so the representation worth learning for planning is value-predictive, not state-predictive. Offline goal reaching makes this actionable. Because the relabeled reward is sparse, the Monte Carlo return from any state to any future goal on its trajectory is a closed-form function of their temporal distance. Value learning therefore becomes regression rather than temporal-difference bootstrapping, and that regression target can supervise the latent dynamics directly, acting as an implicit regularizer against representation collapse. A model trained this way may be a poor simulator and still be a good planner.

Motivated by this view, we introduce **Value-Predictive Flow (VPFLOW)**, which plans offline goal-reaching tasks with a deliberately imperfect world model. VPFLOW learns a latent encoder, a goal-agnostic latent dynamics model, twin goal-conditioned value heads, and a flow-matching action-chunk policy. Imagined latents are trained to be *value-predictive*: rolled forward with dataset actions, the value predicted at each imagined latent must match the closed-form return of the matched future state. No observation is reconstructed, no target latent is matched, and no bootstrapped value network is required. Figure 1 contrasts these designs with ours.

A value-predictive model must still be paired with a planner that proposes feasible actions. Zeroth-order planners such as CEM and MPPI (Rubinstein, 1999; Williams et al., 2017) perturb raw action sequences, which in offline settings quickly leaves the data support and exposes the value function to queries it was never trained on. VPFLOW instead searches in the noise space of its flow policy (Lipman et al., 2023; Park et al., 2025b): each candidate is a sequence in the noise space that the one-step flow decodes into action chunks at the current imagined latent. The zero mean of this space decodes to the policy's own action chunk, so the search warm-starts at multimodal, in-support proposals and every point it visits stays on the data manifold. Our developed optimizer ε-ES, an evolutionary strategy in the noise space, reallocates test-time compute among those proposals by annealed mutation and elite selection under a generalized advantage of the imagined values (Schulman et al., 2016)

On OGBench (Park et al., 2025a) with state or pixel observations, VPFLOW raises mean success from 32.9% to 63.2% over 17 state datasets and from 27.5% to 37.6% over 13 pixel datasets relative to the strongest non-hierarchical baseline in each case. Notably, these results hold while the model's state-prediction error sits orders of magnitude above the state-predictive variant. In fact, we show

that adding the state-predictive term closes that gap at the price of lowering the success rate. This suggests that planning does not necessarily require a faithful simulator, but rather a value-predictive one. Our contributions are as follows:

1. We argue for and provide evidence that value-predictive latent dynamics, trained by closed-form Monte Carlo regression, suffice for offline goal reaching even as state-prediction error grows, challenging the common assumption and objective of learning a (latent) state-predictive world model to plan.
2. We propose a test-time planning algorithm tailored to generative flow policy in its noise space. It steers the policy's initial noise towards a higher generalized advantage, in a way that prevents the out-of-distribution action queries in action space planning.
3. We show on OGBench with state or pixel inputs that VPFLOW surpasses model-free baselines and latent state-predictive planners, and scales with test-time compute on long-horizon tasks.

2 RELATED WORK

We position VPFLOW within three lines of work with broader literature in Appendix A.

Offline goal-conditioned RL. Offline RL learns policy from pre-collected datasets without interacting with the environment, and needs techniques to prevent actions from staying off the data manifold in value learning (Levine et al., 2020; Kostrikov et al., 2022; Kumar et al., 2020; Fujimoto & Gu, 2021; Park et al., 2025b). In the goal-conditioned setting, an agent is tasked to reach any goal from any state (Kaelbling, 1993), and hindsight relabeling (Andrychowicz et al., 2017) can convert any trajectories, even the failure ones, into successful ones with the goal actually reached. With high-coverage datasets, this may enable learning a universal value function that empowers the policy to generalize to unseen goals in zero-shot (Schaul et al., 2015; Park et al., 2024).

Representations for planning with world models. World models seek a compact representation and transition dynamics from observations, where value can be learned in imagination (Sutton, 1991; Ha & Schmidhuber, 2018; Hafner et al., 2020; Hansen et al., 2022). Yet what representations are good for planning is still an open question. Reconstruction-based methods harness the dense supervision from pixels and have achieved remarkable success (Hafner et al., 2019; 2025a). Self-predictive representation holds the promise for a more powerful model (Schwarzer et al., 2021; LeCun et al., 2022; Bagatella et al., 2026), but needs stop gradient, EMA, or regularization to prevent collapse (Sobal et al., 2025; Maes et al., 2026). More critically, it fails to exploit the rich temporal distance supervision naturally from goal reaching. This makes its usefulness for planning not directly obvious. Our work aims to reveal what is essential and necessary for planning-oriented representations.

Test-time search. With a learned world model, previous studies primarily plan actions by sampling-based optimization (Williams et al., 2017; Janner et al., 2019; Hafner et al., 2019; Hansen et al., 2022; Park et al., 2026a) or search (Schrittwieser et al., 2020). Yet action space planning may suffer heavily in the offline setting, especially with low-coverage data, in that off-manifold actions severely undermines. Works with generative policy have turned to steering the noise at test time to keep actions in-distribution; but they need reward or other external signals and have yet to show scaling effectiveness (Yang et al., 2024; Wagenmaker et al., 2025; Patil et al., 2026). In contrast, we show that VPFLOW can improve itself at test time by noising steering [1] with imagined rollouts, even with imperfect world models.

3 PROBLEM STATEMENT AND PRELIMINARIES

Offline goal reaching. We consider the offline goal-conditioned reinforcement learning problem as a reward-free controlled Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \gamma, P)$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space, $\gamma \in [0, 1)$ denotes the discount factor, and $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ denotes the transition dynamics mapping state and action to another state distribution $\Delta(\cdot)$. Given a

[1] We will use "steering", "planning", and "searching" interchangeably in this paper.

dataset $\mathcal{D} = \{\tau^{(i)}\}_{i=1}^{N}$ with trajectories $\tau^{(i)} = (s_0^{(i)}, a_0^{(i)}, s_1^{(i)}, \ldots, s_{T_i-1}^{(i)})$[2] collected by a behavior policy, the objective is to learn a goal-conditioned policy $\pi(a \mid s, g)$ that maximizes the expected utility: $\mathbb{E}_\tau[\sum_{t=0}^{T-1} \gamma^t \delta_g(s_t)]$ with any goal $g \in \mathcal{S}$. Here, the $\delta_g(s_t)$ has been referred to as "Dirac delta function" (Park et al., 2025a). In practice, we consider it as the indicator function $\mathbb{1}_{\{s_t=g\}}$ that represents the relabeled sparse reward:

$$r(s_t, g) = \begin{cases} 1 & s_t = g, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Model predictive control. Given a transition dynamics model $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, model predictive control (MPC) addresses an online constrained optimal control problem with a lookahead action sequence $a_{t:t+H-1}$ at the current state s_t, then executes the first action, and re-plans at the next state. More formally, MPC solves the following optimization problem at time step t:

$$\min_{a_{t:t+H-1}} \sum_{h=0}^{H-1} \ell_t(s_{t+h}, a_{t+h}) + \ell_f(s_{t+H}), \quad \text{s.t.} \quad s_{t+1} = f(s_t, a_t), \tag{2}$$

where $\ell_t(s, a)$ and $\ell_f(s)$ are the cost functions of interest. After finding the minimizer $a^\star_{t:t+H-1}$ of Equation 2, only the first action $a_t^\star$ is applied to handle the compounding error in the dynamics.

In the model-based RL context, the cost has been instantiated by reward $r_t(s_t, a_t)$ and value function $Q(s_t, a_t)$ or $V(s_t)$. Both are learned as part of the world model $f(s_t, a_t)$, which can be used to plan without interacting with the real environment (Hafner et al., 2019; 2020; Hansen et al., 2022). Planning is widely done by sampling-based trajectory optimizer (Rubinstein, 1999; Williams et al., 2017). Another emerging way is to learn planning-enabled representations without the reward and value and use simple ℓ_2 distance to goal as the cost (Assran et al., 2025; Sobal et al., 2025; Maes et al., 2026). Either line of work assumes the optimal objective is to predict the next state when learning world models. In this work, we provide evidence for another value-predictive representation view.

4 VPFlow: Planning with Imperfect World Models

4.1 Value Equivalence Principle

We motivate our method by the value equivalence principle (Grimm et al., 2020) in world modeling. It states that two groups of *reward and dynamics models* are value-equivalent if they yield the identical Bellman operator, meaning that we can learn another set of dynamics and reward beyond the training examples but still keep the same effect for policy learning.

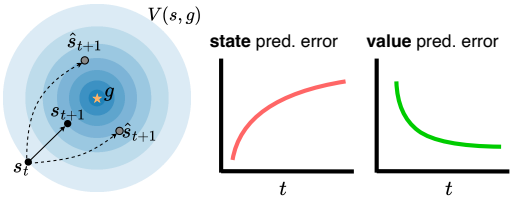


Figure 2: **The value equivalence principle view of world modeling.** The predictive representations of the same timestep constitute a level set of the value field.

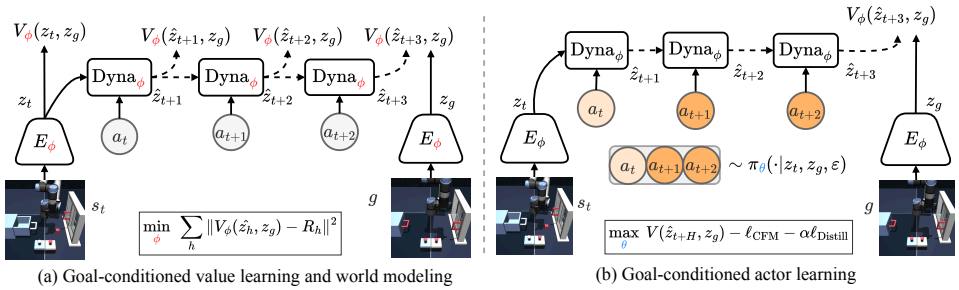
We extend this view to the context of latent world modeling for offline goal reaching, in the sense that two *predictive representations* are value-equivalent if they are scored equally by the goal-conditioned value function. As such, the predictive representations of next state at the same timestep forms a level set of a value function, as shown in Figure 2.

In this setting, the Bellman operator can be written in closed-form with sparse reward given a goal state, reducing the online recursive temporal-difference learning to a regression on a fixed value, unlike value prediction network (Oh et al., 2017). More intuitively, it expands the distribution coverage of latent space, thereby allowing larger search space for policy learning.

4.2 Hindsight Monte Carlo Return Supervision

Q-learning typically uses bootstrapping that predicts Bellman backup online with target network and EMA to learn the action-value function $Q(s, a)$. Though this heuristic stabilizes training (Tang et al., 2023), it makes TD learning less principled for scaling on larger datasets (Park et al., 2026b).

[2]No assumption on its optimality.



(a) Goal-conditioned value learning and world modeling (b) Goal-conditioned actor learning

Figure 3: **VPFLOW objectives.** VPFlow alternatively trains the goal-conditioned value function with predictive dynamics models and a one-step flow policy. The value function regresses the closed-form Monte Carlo returns using imagined rollouts. The policy is improved by maximizing the imagined value from the learned dynamics model, regularized by conditional flow matching to stay close to the data manifold.

With hindsight relabeling, every trajectory in the dataset can be relabeled with a goal as if the agent in that trajectory have reached that goal. This naturally admits a cleaner and simpler objective for value learning. Concretely, the target value for a state s_t is the discounted Monte Carlo return from s_t to the relabeled goal $g = s_{t+d}$ at the same trajectory:

$$R(s_t, g) = R(d) = \sum_{k=0}^{d} \gamma^k r(s_{t+k}, g) = \begin{cases} \gamma^d & r(s,g) = \mathbb{1}_{\{s=g\}}, \\ \frac{1-\gamma^d}{\gamma-1} & r(s,g) = \mathbb{1}_{\{s=g\}} - 1, \end{cases} \tag{3}$$

where $d \geq 0$ is the temporal distance, *i.e.*, the number of steps, to the goal and the full horizon of that episode. In practice, we use the sparse reward $r(s, g) = \mathbb{1}_{\{s=g\}} - 1$, following (Intelligence et al., 2025; Park et al., 2025a). Intuitively, this quantity measures how far the agent is from the goal as a progress signal.

The training of VPFLOW will alternate between the value prediction and policy learning objectives, as shown in Figure 3 and will be elaborated below.

4.3 VALUE-PREDICTIVE LATENT DYNAMICS

Given this closed-form Monte Carlo return in Equation 3, VPFLOW learns the observation encoder $E_\phi : \mathcal{S} \to \mathcal{Z}$, latent dynamics $f_\phi : \mathcal{Z} \times \mathcal{A} \to \mathcal{Z}$, and a goal-conditioned value function $V_\phi : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ by jointly optimizing them with the value-predictive objective. In particular, we use the action chunk representation of horizon H as $\boldsymbol{a}_t = a_{t:t+H-1}$, which allows for a certain level of temporal abstraction (Li et al., 2025). After encoding the state and goal with E_ϕ:

$$z_t = E_\phi(s_t), \quad z_g = E_\phi(g), \tag{4}$$

the latent dynamics predicts next state with the action chunk:

$$\hat{z}_{t+1} = f_\phi(z_t, a_t), \tag{5}$$

and the value-prediction objective with explicit next-state supervision reads:

$$\mathcal{L}_{\text{VP}}(\phi) \coloneqq \mathbb{E}_{(s_t, \boldsymbol{a}_t)\sim\mathcal{D}, g\sim\mathcal{S}} \left[\frac{1}{H+1} \sum_{h=0}^{H} M_{t+h} \odot \|V_\phi(\hat{z}_{t+h}, z_g) - R(s_{t+h}, g)\|^2 \right], \quad \hat{z}_t \coloneqq z_t, \tag{6}$$

where $\odot$ is the Hadamard product, and the mask M_t filters out goals behind time t and across trajectories such that the value function only exploits the clean temporal distance signal on the same trajectory as the sampled state.

Equation 6 trains the encoder, latent dynamics, and value function jointly. The value supervision nudges the predictive representation of each rollout horizon towards the level set of the value function defined by the state-goal temporal distance, instead of aiming for perfect state prediction. It structures the representation space with explicit meaning as progress towards reaching the goal, unlike the regularizers that are only statistically meaningful (Bardes et al., 2022; Balestriero & LeCun, 2025). It also expands the hypothesis space of policy without being too conservative inside the data manifold, as we discussed in Section 4.1.

Algorithm 1: ε-ES: gradient-free noise steering with generalized advantage estimation

Input: state s_t, goal g; number of action chunks n_c; population size N; number of elites E; number of generations K; mutation scales $\sigma_{\max}, \sigma_{\min}$; decay factor ρ; temperature β; receding horizon H_{exec}
Output: the next H_{exec} actions to execute

$z_t \leftarrow E_\phi(s_t),\ z_g \leftarrow E_\phi(g)$;
function $\mathrm{Dec}(\boldsymbol{\varepsilon})$: // closed-loop decoding of n_c chunks
  $z \leftarrow z_t$;
  for $c = 0, \dots, n_c - 1$ **do**
    $\boldsymbol{a}_c \leftarrow \pi_\theta(z, z_g, \varepsilon_c),\ z \leftarrow f_\phi(z, \boldsymbol{a}_c)$;
  return $(\boldsymbol{a}_0, \dots, \boldsymbol{a}_{n_c-1})$;
if $K = 0$ **then return** the first H_{exec} actions of $\mathrm{Dec}(\mathbf{0})$ // output actions directly; no steering;
$\boldsymbol{\varepsilon}^{(e)} \leftarrow \mathbf{0}, A_e \leftarrow A(\mathrm{Dec}(\mathbf{0}); z_t, z_g)$ for $e = 1, \dots, E$; // elites start at the policy mode
for $k = 1, \dots, K$ **do**
  $\sigma_k \leftarrow \max(\sigma_{\min}, \sigma_{\max}\rho^{\,k-1})$;
  for $e = 1, \dots, E$ **do** // $\lceil N/E \rceil$ children per elite
    $\tilde{\boldsymbol{\varepsilon}}_j \sim \mathcal{N}(\boldsymbol{\varepsilon}^{(e)}, \sigma_k^2 I), j = 1, \dots, \lceil N/E \rceil$;
    $\tilde{A}_j \leftarrow A(\mathrm{Dec}(\tilde{\boldsymbol{\varepsilon}}_j); z_t, z_g), j^+ \leftarrow \arg\max_j \tilde{A}_j$; // Equation 10
    if $\tilde{A}_{j^+} > A_e$ **then** $(\boldsymbol{\varepsilon}^{(e)}, A_e) \leftarrow (\tilde{\boldsymbol{\varepsilon}}_{j^+}, \tilde{A}_{j^+})$;
$e^\star \sim p(e) \propto \exp(\beta A_e)$;
return the first H_{exec} actions of $\mathrm{Dec}(\boldsymbol{\varepsilon}^{(e^\star)})$ // execute, then re-plan at the state reached

4.4 FLOW POLICY AS A BEHAVIOR PRIOR

For a more expressive policy to handle the multi-modality of offline datasets, we opt for a flow-based policy similar to Flow Q-learning (FQL) (Park et al., 2025b). Beyond that, the generative policy as a behavior prior is also a key design choice to enable planning with the data manifold constraint at test time, as we will discuss in Section 4.5.

The policy improvement objective will be to maximize the Q function with action chunk (Li et al., 2025) with encoded state and goal, $z_t = E_\phi(s_t), z_g = E_\phi(g)$, using the learned world models:

$$\begin{aligned} Q(z_t, \boldsymbol{a}_t, z_g) &\coloneqq \mathbb{E}_{\hat{z}_{t+H} \sim f_\phi(z_t, \boldsymbol{a}_t)} \left[\sum_{h=0}^{H-1} \gamma^h r(z_{t+h}, z_g) + \gamma^H V_\phi(\hat{z}_{t+H}, z_g) \right], \\ &\approx \mathbb{E}_{\hat{z}_{t+H} \sim f_\phi(z_t, \boldsymbol{a}_t)} \left[\gamma^H V_\phi(\hat{z}_{t+H}, z_g) \right] \qquad \text{(no } \theta \text{ in reward)}. \end{aligned} \tag{7}$$

where $\boldsymbol{a}_t = a_{t:t+H-1}$ is the output from the one-step flow policy $\pi_\theta(z_t, z_g, \varepsilon_t)$. With a velocity field v_θ, the policy learning loss reads:

$$\mathcal{L}_\pi(\theta) \coloneqq -Q(z_t, \boldsymbol{a}_t, z_g) + \alpha_{\text{CFM}}\mathcal{L}_{\text{CFM}} + \alpha_{\text{Distill}}\mathcal{L}_{\text{Distill}}. \tag{8}$$

This one-step flow policy $\pi_\theta(z, z_g, \varepsilon)$ distills the multi-step decoding process using velocity $v_\theta(z, z_g)$ that makes noise steering efficient. Detailed objective functions will be shown in Appendix B.3.

4.5 STEERING THE NOISE AS PLANNING

Planning objective. VPFLOW instantiates the cost objective in Equation 2 as the reward function r_t defined in Equation 1, and optimize the action chunk $\boldsymbol{a}_t = a_{t:t+H-1}$ at state z_t. Yet, credit assignment is challenging for sparse reward. As such, we densify it with the potential-based reward shaping (Ng et al., 1999):

$$\delta_t = r_t + \gamma V_\phi(\hat{z}_{t+1}, z_g) - V_\phi(z_t, z_g). \tag{9}$$

We design the planning objective to maximize the discounted sum of the reshaped reward for the planning horizon H as:

$$A_t = \sum_{h=0}^{H-1} (\gamma\lambda)^h \delta_{t+h}, \quad \text{s.t.} \quad \hat{z}_{t+1} = f_\phi(z_t, a_t). \tag{10}$$

This corresponds to the Generalized Advantage Estimator (GAE) (Schulman et al., 2016). It measures how much better, in our sense, how near the rollout state is to the goal, compared to the current state.

Noise space steering. Planning is widely conducted by directly optimizing actions on an unconstrained support. For long-horizon rollouts, actions tend to drift away from the data distribution,

Table 1: **Success rate (%, ↑) on OGBench compared against model-free baselines.** The results are averaged over the datasets of each environment by their mean, with the number of datasets in parentheses. The best and second best results in each row are in **bold** and underlined, respectively.

Input	Task Type	Environment	GCBC	GCIVL	GCIQL	QRL	CRL	VPFLOW
State	Locomotion	PointMaze (4)	16.0	38.2	27.8	60.0	29.8	**87.8**
		AntMaze (4)	19.8	31.8	35.0	53.0	61.8	**68.0**
	Manipulation	Cube (4)	2.0	22.5	27.8	1.5	8.2	**42.2**
		Scene (1)	5.0	42.0	51.0	5.0	19.0	**78.0**
		Puzzle (4)	0.5	9.0	36.8	0.2	2.0	**51.0**
		Overall (17)	9.3	26.4	32.9	27.3	25.1	**63.2**
Pixel	Locomotion	AntMaze (4)	5.0	9.0	5.2	1.5	**68.2**	61.0
	Manipulation	Cube (4)	7.2	21.0	13.2	16.8	13.5	**34.8**
		Scene (1)	12.0	25.0	12.0	10.0	11.0	**33.0**
		Puzzle (4)	4.2	**28.2**	6.5	0.2	4.8	18.2
		Overall (13)	6.0	19.8	8.6	6.5	27.5	**37.6**

making the planning objective unreliable. One of the key benefits of generative policy is that after being trained, its noise space can be optimized to search for better actions while still being close to the training distribution (Yang et al., 2024; Wagenmaker et al., 2025). We adopt this strategy to steer the noise to optimize Equation 10.

ε-ES: evolutionary search in noise space CEM planner used in model-based methods maintain an elite set of actions, from which an online mean and standard deviation are updated (Janner et al., 2019; Hansen et al., 2022; Maes et al., 2026). However, that design choice does not directly translate to the noise space, as their distribution differs. Inspired by a previous gradient-free optimizer for diffusion noise (Yang et al., 2024), we instead adopt an evolution strategy that mutates samples of each elite in the noise space and greedily keeps the ones that score higher by the objective in Equation 10. We present the search process in Algorithm 1 with more details in Appendix B.7.

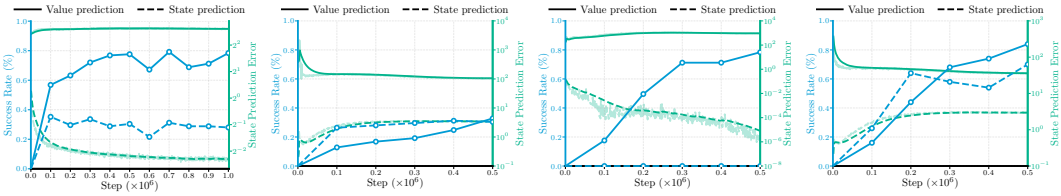
5 EXPERIMENTS

We aim to address four questions in this section: **Q1**: How does VPFLOW compare with model-free and model-based methods that similarly plan with latent world models for offline goal-reaching tasks? **Q2**: Are state-predictive representations useful for planning, or do the value-predictive ones play a more essential role? **Q3**: Does test-time steering help, and does the search space matter more than the search algorithm? **Q4**: How does the learned latent actually behave during rollouts?

Setup. We use the official OGBench datasets and tasks (Park et al., 2025a): 17 state-based datasets and 13 pixel-based datasets. The metric is the goal-reaching success rate (%, ↑), averaged over the five tasks within each dataset. Model-free baselines are the non-hierarchical methods reported by OGBench: **GCBC** (Ghosh et al., 2021), **GCIVL** and **GCIQL** (Kostrikov et al., 2022), **QRL** (Wang et al., 2023), and **CRL** (Eysenbach et al., 2022). Model-based baselines are the latent planners **LeWorldModel** (Maes et al., 2026), **PLDM** (Sobal et al., 2025), and **TD-MPC2** (Hansen et al., 2024). Implementation details are in Appendix B, and hyperparameters are in Appendix D.

5.1 Q1: OFFLINE GOAL REACHING ON OGBENCH

Comparison against model-free baselines. Table 1 groups the results by environment (full numerics in Appendix E). Averaged over the 17 state datasets, VPFLOW nearly doubles the strongest baseline (63.2 vs. 32.9 for GCIQL) and leads every environment group. The gains concentrate on manipulation (Scene 78.0 vs. 51.0, Puzzle 51.0 vs. 36.8) and shrink on navigation (AntMaze 68.0 vs. 61.8), where a reactive policy already solves much of the task. This suggests that model-based planning can be more suitable for contact-rich tasks, where test-time compute translates more directly to success. Meanwhile, the pixel group shows modest gains (37.6 vs. 27.5 for CRL), limited by two datasets.



(a) `scene-play-v0` (b) `visual-scene-play-v0` (c) `visual-cube-single-play-v0` (d) `visual-antmaze-large-navigate-v0`

Figure 5: **Comparison of value and state-predictive representations regarding success rate and next-state prediction error.** On state and pixel-based inputs, the value-predictive representation achieves higher success even when its state prediction error is a magnitude larger.

Table 2: **Ablations on steering algorithm and search space.** Action space planning worsen the success rate from direct action output.

Planner	Space	Loco.	Manip.	All
w/o steering	–	39.4	8.3	28.0
w/ $\mathcal{A}$-CEM	action	22.6	1.6	10.0
w/ ε-CEM	noise	53.2	31.5	40.2
VPFlow	noise	**72.2**	**38.7**	**52.1**

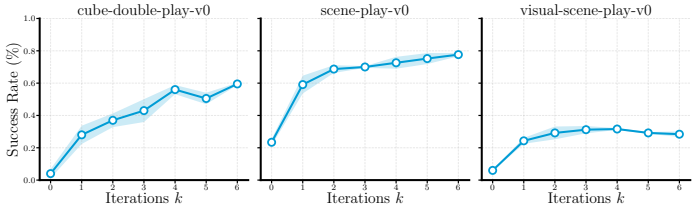


Figure 6: **Success rate against the number of iterations** K. More iterations steer the policy towards higher success. $K = 0$ means reactive policy without steering.

Comparison against latent planning methods. To our surprise, on 12 of the 13 pixel datasets, *all three model-based baselines score zero*; so Figure 4 reports the only dataset on which any one of them achieves non-zero success. All three train their latent to predict a future state, then plan over raw action sequences, and on reward-free play data neither practices stand out as the optimal design choices. We will next reveal more about the possible failure modes on these two patterns.

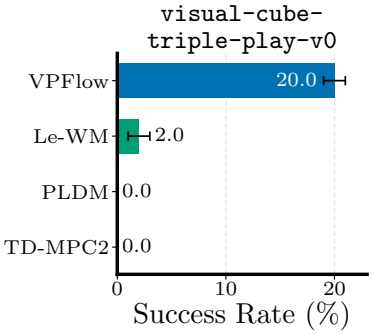


Figure 4: **Planners with latent world models**, on *the only* dataset where other baselines do not fail completely.

5.2 Q2: VALUE-PREDICTIVE OR STATE-PREDICTIVE LATENTS?

This experiment changes one thing. The default model trains the rollout with $\mathcal{L}_{\text{VP}}$ (Equation 6); the state-predictive control adds the latent-matching term (Equation 14), and other configurations stay the same. Figure 5 tracks success and the next-latent prediction error during training on four datasets.

Lower prediction error, worse control. The state-predictive controller trained with Equation 14 reduces the next-latent error by one to eight orders of magnitude: at the end of training the error is 0.21 against 6.07 (Figure 5a), 3.5 vs. 106 (Figure 5b), 2.9 vs. 35.7 (Figure 5d), and 4.9×10^{-6} vs. 948 (Figure 5c). Success moves on the same direction, reducing from 78.4% to 28.0% , 32.8% to 30.4%, 84.0% to 70.0%, and 78.4% to 0.0%! A model that predicts its own next latent almost perfectly is therefore not a model that plans well. Predicting the value is more essential for planning.

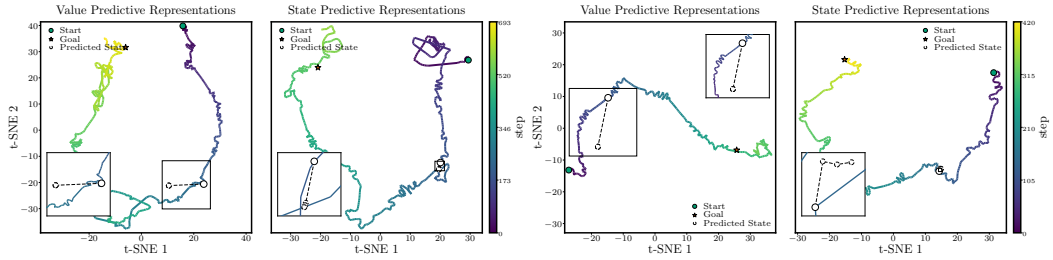
The failure mode is collapse, and value prediction prevents it. On `visual-cube-single` (Figure 5c), the latent-matching term collapses to near zero: the encoder finds a trivially predictable latent and the agent never reaches the goal, the degeneracy that self-predictive objectives prevents through explicit anti-collapse techniques, none of which our architecture has. In contrast, Equation 6 promotes temporally ordered latents that implicitly prevent collapse.

5.3 Q3: Does steering help, and where should it search?

Test-time steering enables policy improvement. As shown in Table 2, VPFlow without steering (K=0) executes actions directly and achieves inferior performance. Extra test-time compute by updating noise greedily towards maximal advantage in the imagined future greatly improves success. Figure 6 traces the gain per iteration of K: success rises with more iterations. This supports Equation 10 as an effective objective enabling policy improvement via search without external signal.

Noise space sterering proves more effective in action space. Replacing ε-ES with ε-CEM that updates noise with standard CEM lowers the success rate. This is expected as noise is updated by advantage on the decoded actions rather than the noise itself. More notably, the success of action CEM is even worse than its direct output (10.0 vs. 28.3) and not even close to its noise counterpart (vs. 40.2). This can be attributed to the advantage exploitation of off-manifold actions, which is prevented in noise space because its decoded actions are trained to be in the support of value function.

5.4 Q4: How does the latent behave during an episode?



(a) `visual-antmaze-large-navigate-v0` Task1 (b) `visual-antmaze-large-navigate-v0` Task3

Figure 7: **t-SNE of latent trajectories from successful episodes on Visual AntMaze.** The solid circle is the encoded current state, and the dashed circles are the imagined latents via dynamics for that step. Under value-predictive training, the imagined latents leave the trajectory manifold, but the agent can still reach the goal.

Section 5.2 showed that a large next-state error is compatible with good control; Figure 7 elucidates it visually. The state-predictive imagined latent stays on the state manifold, which is what its objective enforces; the value-predictive one falls well off it, yet the agent still reaches the goal. The imagined latent is therefore not a state estimate: f_ϕ moves the representations to where V_ϕ forecasts the right goal-reaching value, regardless of any reachable observation encoded into it. This also explains the objective's tolerance of play data: value regression needs to order futures by their distance to goal, so it can quotient out irrelevant variation to reach the goal, while a state-predictive model must account for every sample in the demonstrations.

6 Conclusions

We have challenged the assumption that planning requires a state-predictive world model. VPFlow trains latent dynamics purely through the closed-form Monte Carlo returns that offline goal reaching provides for free, and acts by steering a distilled flow policy inside its own noise space. On OGBench with state and pixel inputs, VPFlow surpasses model-free baselines and latent state-predictive planners while its next-state prediction error stays orders of magnitude above a state-predictive control; closing that gap lowers success throughout. Steering converts the behavior prior into those results, and the coordinate system matters more than the search algorithm. The broader lesson is that a controller needs value prediction from its model and a coordinate system from its policy, rather than a faithful simulator. This perspective extends beyond goal reaching: where a value target is available in closed form, simulation fidelity may be exchanged for control relevance, and the noise-space coordinate system transfers to other generative policies with a searchable source. Limitations and future directions are in Appendix F.

AI USE STATEMENT

In this work, we used generative AI tools for **design or provide feedback on research methodology or experiments,implement methods, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis,and interpret results**. We have not used generative AI tools for **generate synthetic data sets, help develop theoretical models or conceptual frameworks,formulate mathematical claims, provide critical ingredients for proving mathematical claims,assist in the writing of proofs, and propose or refine hypotheses**, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for [create or modify scientific figures or images,draft parts of a research paper]. We have reviewed all AI-assisted work. **We checked LLM-generated research ideas for potential plagiarism through a manual literature survey. LLM-generated code was verified and tested for correctness by authors**. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

All experiments use the public OGBench datasets and evaluation tasks (Park et al., 2025a); Appendix B specifies the datasets, goal relabeling scheme, network architectures, objectives, optimizer, training schedule, training pseudocode, and evaluation protocol, and Appendix D lists every training and steering hyperparameter, including the search grid and the per-environment settings used in the reported tables. Appendix C derives the closed-form Monte Carlo target, the GAE steering score, and the first-order update, and Appendix E reports per-dataset results with standard deviations across seeds together with the search-coordinate control.

REFERENCES

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017. 1, 3, 16

Arthur Argenson and Gabriel Dulac-Arnold. Model-based offline planning. In *International Conference on Learning Representations*, 2021. 16

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025. 4

Marco Bagatella, Matteo Pirotta, Ahmed Touati, Alessandro Lazaric, and Andrea Tirinzoni. Td-jepa: Latent-predictive representations for zero-shot reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 36211–36244, 2026. 3, 16

Randall Balestriero and Yann LeCun. Lejepa: Provable and scalable self-supervised learning without the heuristics. *arXiv preprint arXiv:2511.08544*, 2025. 5

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations*, 2022. 5

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado Van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30, 2017. 2, 16

Andre Barreto, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel Mankowitz, Augustin Zidek, and Remi Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *International conference on machine learning*, pp. 501–510. PMLR, 2018. 16

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. 18

Léonard Blier, Corentin Tallec, and Yann Ollivier. Learning successor states and goal-dependent values: A mathematical viewpoint. *arXiv preprint arXiv:2101.07123*, 2021. 16

Diana Borsa, Andre Barreto, John Quan, Daniel J. Mankowitz, Hado van Hasselt, Remi Munos, David Silver, and Tom Schaul. Universal successor features approximators. In *International Conference on Learning Representations*, 2019. 16

Wilka Carvalho, Momchil S Tomov, William de Cothi, Caswell Barry, and Samuel J Gershman. Predictive representations: Building blocks of intelligence. *Neural Computation*, 36(11):2225–2298, 2024. 16

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural computation*, 5(4):613–624, 1993. 2, 16

Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472, 2011. 16

Yilun Du, Toru Lin, and Igor Mordatch. Model-based planning with energy-based models. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 374–383. PMLR, 30 Oct–01 Nov 2020. 16

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pp. 1407–1416. PMLR, 2018. 17

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35:35603–35620, 2022. 1, 7, 16

Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 2786–2793. IEEE, 2017. 16

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021. 3, 16

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019. 25

Scott Fujimoto, Pierluca D'Oro, Amy Zhang, Yuandong Tian, and Michael Rabbat. Towards general-purpose model-free reinforcement learning. In *International Conference on Learning Representations*, volume 2025, pp. 83509–83552, 2025. 16

Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Manon Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. In *International Conference on Learning Representations*, 2021. 7, 16

Christopher Grimm, André Barreto, Satinder Singh, and David Silver. The value equivalence principle for model-based reinforcement learning. *Advances in neural information processing systems*, 33: 5541–5552, 2020. 2, 4, 16

Zhaohan Guo, Shantanu Thakoor, Miruna Pîslar, Bernardo Avila Pires, Florent Altché, Corentin Tallec, Alaa Saade, Daniele Calandriello, Jean-Bastien Grill, Yunhao Tang, et al. Byol-explore: Exploration by bootstrapped prediction. *Advances in neural information processing systems*, 35: 31855–31870, 2022. 16

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3):440, 2018. 3, 16

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019. 1, 3, 4, 16

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. 1, 3, 4, 16

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025a. 3, 16

Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models. *arXiv preprint arXiv:2509.24527*, 2025b. 16

Nick Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, volume 2024, pp. 47376–47405, 2024. 1, 7, 16

Nicklas A Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 8387–8406. PMLR, 17–23 Jul 2022. 1, 3, 4, 7, 16

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. $\pi^*_{0.6}$: A VLA that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025. 5

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019. 3, 7, 16

Leslie Pack Kaelbling. Learning to achieve goals. In *IJCAI*, volume 2, pp. 1094–8, 1993. 1, 3, 16

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33: 21810–21823, 2020. 16, 24

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. 21, 24

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022. 1, 3, 7, 16

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020. 3, 16

Yann LeCun et al. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62, 2022. 3, 16

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020. 1, 3, 16

Jinning Li, Chen Tang, Masayoshi Tomizuka, and Wei Zhan. Hierarchical planning through goal-conditioned offline reinforcement learning. *IEEE Robotics and Automation Letters*, 7(4):10216–10223, 2022. 16

Qiyang Li, Zhiyuan (Paul) Zhou, and Sergey Levine. Reinforcement learning with action chunking. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 55518–55553. Curran Associates, Inc., 2025. doi: 10.52202/085713-1856. 5, 6, 16

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. 2, 18

Michael Littman and Richard S Sutton. Predictive representations of state. *Advances in neural information processing systems*, 14, 2001. 16

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Conference on robot learning*, pp. 1113–1132. Pmlr, 2020. 16

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworld-model: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026. 1, 3, 4, 7, 17

Vivek Myers, Bill Zheng, Benjamin Eysenbach, and Sergey Levine. Offline goal-conditioned reinforcement learning with quasimetric representations. In *Advances in Neural Information Processing Systems*, 2025. 1, 16

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287. Citeseer, 1999. 6

Tianwei Ni, Benjamin Eysenbach, Erfan Seyedsalehi, Michel Ma, Clement Gehring, Aditya Mahajan, and Pierre-Luc Bacon. Bridging state and history representations: Understanding self-predictive rl. In *International Conference on Learning Representations*, volume 2024, pp. 23555–23569, 2024. 16

Junhyuk Oh, Satinder Singh, and Honglak Lee. Value prediction network. *Advances in neural information processing systems*, 30, 2017. 2, 4, 16

Kwanyoung Park, Seohong Park, Youngwoon Lee, and Sergey Levine. Scalable offline model-based rl with action chunks. In *International Conference on Learning Representations*, volume 2026, pp. 9599–9623, 2026a. 3, 16, 25

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. Hiql: Offline goal-conditioned rl with latent states as actions. *Advances in Neural Information Processing Systems*, 36:34866–34891, 2023. 16

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with hilbert representations. In *Forty-first International Conference on Machine Learning*, 2024. 3, 16

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations*, volume 2025, pp. 94937–94982, 2025a. 1, 2, 4, 5, 7, 10, 16, 17, 19, 20, 23

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *Forty-second International Conference on Machine Learning*, 2025b. 2, 3, 6, 16, 25

Seohong Park, Kevin Frans, Deepinder Mann, Benjamin Eysenbach, Aviral Kumar, and Sergey Levine. Horizon reduction makes rl scalable. *Advances in Neural Information Processing Systems*, 38:8350–8389, 2026b. 4

Omkar Patil, Ondrej Biza, Thomas Weng, Karl Schmeckpeper, Wil Thomason, Xiaohan Zhang, Kausik Sivakumar, Robin Walters, Nakul Gopalan, Sebastian Castro, et al. You've got a golden ticket: Improving generative robot policies with a single noise vector. *arXiv preprint arXiv:2603.15757*, 2026. 3

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019. 16

Matteo Pirotta, Andrea Tirinzoni, Ahmed Touati, Alessandro Lazaric, and Yann Ollivier. Fast imitation via behavior foundation models. In *International Conference on Learning Representations*, volume 2024, pp. 12685–12724, 2024. 16

Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1(2):127–190, 1999. 2, 4, 16

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pp. 1312–1320. PMLR, 2015. 1, 2, 3, 16

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020. 1, 3, 16

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. 2016. 2, 6

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2021. 1, 3, 16

Uladzislau Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim G. J. Rudner, and Yann LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. In *Advances in Neural Information Processing Systems*, 2025. 1, 3, 4, 7, 17

Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991. 1, 3, 16

Yunhao Tang, Zhaohan Daniel Guo, Pierre Harvey Richemond, Bernardo Avila Pires, Yash Chandak, Rémi Munos, Mark Rowland, Mohammad Gheshlaghi Azar, Charline Le Lan, Clare Lyle, et al. Understanding self-predictive learning for reinforcement learning. In *International conference on machine learning*, pp. 33632–33656. PMLR, 2023. 4, 16

Stephen Tian, Suraj Nair, Frederik Ebert, Sudeep Dasari, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. Model-based visual planning with self-supervised functional distances. In *International Conference on Learning Representations*, 2021. 16

Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. *Advances in Neural Information Processing Systems*, 34:13–23, 2021. 16

Hugues Van Assel, Mark Ibrahim, Tommaso Biancalani, Aviv Regev, and Randall Balestriero. Joint-embedding vs reconstruction: Provable benefits of latent space prediction for self-supervised learning. In *Advances in Neural Information Processing Systems*, 2025. 16

Andrew Wagenmaker, Yunchu Zhang, Mitsuhiko Nakamoto, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. In *9th Annual Conference on Robot Learning*, 2025. 3, 7

Mianchu Wang, Rui Yang, Xi Chen, Hao Sun, Meng Fang, and Giovanni Montana. GOPlan: Goal-conditioned offline reinforcement learning by planning with learned models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. 16

Tongzhou Wang, Antonio Torralba, Phillip Isola, and Amy Zhang. Optimal goal-reaching reinforcement learning via quasimetric learning. In *International Conference on Machine Learning*, pp. 36411–36430. PMLR, 2023. 1, 7, 16

Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. *Advances in neural information processing systems*, 28, 2015. 16

Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2): 344–357, 2017. 2, 3, 4, 16

Brian Yang, Huangyuan Su, Nikolaos Gkanatsios, Tsung-Wei Ke, Ayush Jain, Jeff Schneider, and Katerina Fragkiadaki. Diffusion-es: Gradient-free planning with diffusion for autonomous and instruction-guided driving. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15342–15353. IEEE, 2024. 3, 7

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in neural information processing systems*, 33:14129–14142, 2020. 16, 24

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021. 16

Wancong Zhang, Basile Terver, Artem Zholus, Soham Chitnis, Harsh Sutaria, Mido Assran, Randall Balestriero, Amir Bar, Adrien Bardes, Yann LeCun, et al. Hierarchical planning with latent world models. *arXiv preprint arXiv:2604.03208*, 2026. 16

A Extended related work

This appendix expands the three paragraphs of Section 2 with the surrounding literature.

A.1 Offline and goal-conditioned reinforcement learning

Offline RL learns from a fixed dataset and must control distribution shift (Levine et al., 2020), either by constraining the actor toward the data (Peng et al., 2019; Fujimoto & Gu, 2021) or by regularizing the critic (Kumar et al., 2020; Kostrikov et al., 2022). Flow Q-learning replaces the Gaussian actor with a flow-matching policy distilled into a one-step network (Park et al., 2025b), which is the actor VPFlow adopts along with action chunks representations (Li et al., 2025). Goal-conditioned RL builds on goal-reaching formulations (Kaelbling, 1993), universal value function approximators (Schaul et al., 2015), and hindsight relabeling (Andrychowicz et al., 2017). Offline variants add iterated supervised learning (Ghosh et al., 2021), latent plans from play (Lynch et al., 2020), contrastive (Eysenbach et al., 2022) and quasimetric (Wang et al., 2023; Myers et al., 2025) value learning, and hierarchical actors (Park et al., 2023), with OGBench standardizing evaluation (Park et al., 2025a).

A.2 Model-based RL and world models

Model-based RL dates to Dyna (Sutton, 1991) and Gaussian-process policy search (Deisenroth & Rasmussen, 2011). Deep variants learn dynamics from pixels through reconstruction (Watter et al., 2015; Finn & Levine, 2017; Hafner et al., 2019) or in a latent space learned for imagination (Ha & Schmidhuber, 2018; Hafner et al., 2020; 2025a;b), and limit compounding error with short rollouts (Janner et al., 2019; Park et al., 2026a). Offline, MOPO (Yu et al., 2020), MOReL (Kidambi et al., 2020), and COMBO (Yu et al., 2021) penalize or truncate rollouts where the model is uncertain, and MBOP (Argenson & Dulac-Arnold, 2021) plans with an ensemble model and a behavior prior. In all of these, the model predicts the next observation and its planning utility is presumed to follow. Value-equivalent models take the opposite position: value prediction networks and MuZero train the model only through predicted values, rewards, and policies (Oh et al., 2017; Schrittwieser et al., 2020), the value-equivalence principle formalizes when this suffices (Grimm et al., 2020), and TD-MPC brings it to continuous control with a TD-trained latent value (Hansen et al., 2022; 2024). VPFlow belongs to this family and differs in the supervision: closed-form Monte Carlo returns in offline goal reaching replace bootstrapped TD targets, which removes the target value network and makes the regression stationary.

A.3 Predictive representations

Predictive representations encode what a state implies about the future rather than what it looks like (Littman & Sutton, 2001; Carvalho et al., 2024). Successor representations and features (Dayan, 1993; Barreto et al., 2017; 2018; Borsa et al., 2019), forward–backward representations (Touati & Ollivier, 2021; Blier et al., 2021), and the behavior foundation models built on them (Pirotta et al., 2024; Park et al., 2024) predict discounted future occupancy, from which goal-conditioned values follow. Self-predictive objectives instead regress future latents, using target encoders or stop-gradients to avoid collapse (Schwarzer et al., 2021; Guo et al., 2022; Tang et al., 2023; Ni et al., 2024; Fujimoto et al., 2025); TD-JEPA extends this to zero-shot RL (Bagatella et al., 2026), and joint-embedding prediction is argued, and in linear settings proven, to be preferable to reconstruction (LeCun et al., 2022; Van Assel et al., 2025). VPFlow shares the similar latent world models architecture but changes the target from future latents to goal-conditioned values.

A.4 Planning in latent space

CEM and MPPI (Rubinstein, 1999; Williams et al., 2017) are the standard planners in PlaNet, MBPO-style controllers, and TD-MPC (Hafner et al., 2019; Janner et al., 2019; Hansen et al., 2022). Energy-based and functional-distance objectives score imagined futures in visual domains (Du et al., 2020; Tian et al., 2021), hierarchical planners decompose long horizons into subgoals (Li et al., 2022; Zhang et al., 2026), and GOPlan plans offline with a learned model and a generative behavior prior (Wang et al., 2024). Closest to our setting, PLDM and LeWorldModel plan with latent dynamics

learned from reward-free offline data, using VICReg- or SIGReg-style regularizers to keep the latent state-predictive (Sobal et al., 2025; Maes et al., 2026). These controllers perturb raw action sequences and inherit the state-predictive objective. VPFLOW searches in the noise space of a distilled flow policy instead, so the policy both warm-starts and constrains the population, and trains its model for value prediction, so the controller does not depend on a faithful latent simulator.

B IMPLEMENTATION DETAILS

This appendix specifies the datasets, architectures, objectives, and optimization used for all reported VPFLOW results. Numerical hyperparameters are collected in Appendix D.

B.1 DATASETS AND GOAL RELABELING

Benchmark. We use the public OGBench datasets and evaluation tasks (Park et al., 2025a) without modification. The state suite comprises 17 datasets: `pointmaze-{medium, large, giant, teleport}-navigate`, `antmaze-{medium, large, giant, teleport}-navigate`, `cube-{single, double, triple, quadruple}-play`, `scene-play`, and `puzzle-{3x3, 4x4, 4x5, 4x6}-play`. The pixel suite comprises the 13 `visual-` counterparts of the AntMaze, Cube, Scene, and Puzzle datasets, with $64 \times 64 \times 3$ observations. Each dataset has five evaluation tasks specified by start–goal pairs.

Segments and goals. A training example is a length-H segment $(s_t, a_t, \dots, a_{t+H-1})$ drawn uniformly from the dataset trajectories, with $H = 16$. The value goal g^V is sampled from a mixture: the current observation s_t with probability 0.2, a future observation s_{t+d} on the same trajectory with $d \sim \mathrm{Geom}(1-\gamma)$ with probability 0.5, and a uniformly random dataset observation with probability 0.3. For each offset $h \in \{0, \dots, H\}$, the MC target is $R_{t+h} = (1 - \gamma^{d_h})/(\gamma - 1)$ with d_h the number of transitions from s_{t+h} to g^V, and the mask M_{t+h} is one iff g^V is on the same trajectory at or after index $t + h$. Random goals therefore contribute no value target. The actor goal g^π is a future observation on the same trajectory.

Augmentation. For pixel tasks, with probability 0.5 per batch we apply random-crop augmentation (edge-pad by three pixels, then crop back to 64×64), using one shared crop per sample for the root observation, all future observations, and both goals, so that temporal and goal alignment is preserved.

B.2 NETWORKS

Encoder. State tasks use $E_\phi(s) = s$. Pixel tasks use one IMPALA-Small convolutional encoder (Espeholt et al., 2018) shared by observations and goals, followed by a 512-unit MLP head that produces the latent.

Dynamics. f_ϕ is goal-agnostic and maps a root latent and an action sequence to one latent per action prefix. It is a cross-attention Transformer in which the root latent is the query and the embedded actions are keys and values, with model width 192, 3 layers, 3 heads, rotary position embeddings, and dropout 0.1; the prediction for prefix h attends only to $a_{t:t+h}$, so all H prefix latents of a chunk are produced in one parallel call. Writing the parallel form explicitly,

$$\hat{z}_{t+h+1} = f_\phi(z_t, a_{t:t+h}), \qquad h = 0, \dots, H-1, \tag{11}$$

every prefix latent is produced from the same root z_t.

Value heads. Each of the two heads is a three-layer MLP of width 512 with layer normalization and GELU activations acting on the concatenation (z, z_g). Policy improvement and planning use $\min\{V_{\phi,1}, V_{\phi,2}\}$. The default MC objective uses no target value network.

Flow policy. The velocity field v_θ and the distilled one-step decoder π_θ are Transformers with width 192, 3 layers, 3 heads, and sinusoidal position embeddings. Distillation regresses the one-step map onto the integrated endpoint of the online velocity field,

$$\mathcal{L}_{\mathrm{Distill}}(\theta) := \mathbb{E}\Big[\big\|\pi_\theta(z_t, z_{g^\pi}, \varepsilon) - \mathrm{sg}\big(\mathrm{ODE}(v_\theta; \varepsilon)\big)\big\|_2^2\Big], \tag{12}$$

with sg as the stop-gradient operation. Conditioning follows π_0 (Black et al., 2024): the projected latent z and goal z_g are prefix tokens that the H action tokens attend to, and the flow time τ enters through a sinusoidal embedding fused into each action token by a two-layer MLP. Action tokens attend to one another bidirectionally for state tasks and causally (prefix-LM) for pixel tasks. The one-step decoder is trained to match a 10-step Euler integration of the online velocity field (Equation 12); no exponential moving average teacher is used. Actions are continuous and decoded actions are clipped to $[-1, 1]$.

B.3 Objectives

Value-predictive model. The default model minimizes $\mathcal{L}_{\mathrm{VP}}$ in Equation 6, averaged over the batch and normalized by the number of unmasked targets.

Conditional flow matching. Trajectories in behavior dataset $\mathcal{D}$ is multimodal, so the prior cannot be a unimodal Gaussian actor. Conditional flow matching learns a velocity field v_θ that transports a source variable $\varepsilon \sim \mathcal{N}(0, I)$ to a demonstrated action chunk $\boldsymbol{a}$ along the linear path $x_\tau = (1-\tau)\varepsilon + \tau \boldsymbol{a}$, whose target velocity is $\boldsymbol{a} - \varepsilon$ (Lipman et al., 2023):

$$\mathcal{L}_{\mathrm{CFM}}(\theta) := \mathbb{E}_{(s,\boldsymbol{a},g)\sim\mathcal{D},\,\varepsilon,\,\tau}\left[\left\|v_\theta(x_\tau, \tau, E_\phi(s), E_\phi(g)) - (\boldsymbol{a} - \varepsilon)\right\|_2^2\right]. \tag{13}$$

The coefficient α_{CFM} in Equation 8 weights Equation 13 during policy improvement. Integrating v_θ is too slow to call inside a search loop, which is why Equation 12 distills it into the one-step map $\pi_\theta(z, z_g, \varepsilon)$ that Section 4.5 searches over.

Latent-predictive control. The latent-predictive control used in Section 5.2 trains the value heads on encoded dataset states $z_{t+h} = E_\phi(s_{t+h})$ and trains the rollout with $\lambda_{\mathrm{state}} = 1$ to match them:

$$\mathcal{L}_{\mathrm{LP}}(\phi; \lambda_{\mathrm{state}}) := \frac{1}{H+1}\sum_{h=0}^{H} M_{t+h} \odot \left\|V_\phi(z_{t+h}, z_{g^V}) - R_{t+h}\right\|^2 + \lambda_{\mathrm{state}} \sum_{h=1}^{H} \left\|\hat{z}_{t+h} - z_{t+h}\right\|_2^2. \tag{14}$$

The latent term is an unmasked mean over latent dimensions and rollout steps, and z_{t+h} has no stop-gradient operation, so the pixel encoder is updated through this target as in joint-embedding self-predictive objectives. Thus the default model uses Equation 6 and each $\lambda_{\mathrm{state}} > 0$ control uses Equation 14; the control is not an additive regularizer on top of the value-predictive rollout. A probabilistic variant replaces the squared error with a diagonal-Gaussian negative log-likelihood.

Actor. The actor minimizes Equation 8 with the model frozen. Policy improvement imagines one chunk ($H = 16$), and the value term is normalized by the stop-gradient batch mean of $|q_\theta|$ with $\epsilon = 10^{-6}$.

B.4 Training loop

Algorithm 2 gives the loop referenced in Section 4.4. Policy evaluation and policy improvement alternate on the same minibatch, with the model frozen during the actor update, and no target value network appears anywhere in the default configuration.

B.5 Optimization

All networks use Adam with learning rate 2×10^{-4}, cosine decay to zero, batch size 256, and gradient-norm clipping at 10. State runs use 10^6 gradient updates and pixel runs 5×10^5. The model and actor are updated alternately at every step ($M = 1$). Loss coefficients are $\alpha_{\mathrm{CFM}} = 1$ and $\alpha_{\mathrm{Distill}} = 200$ (state) or 50 (pixel).

B.6 Steering and evaluation protocol

Steering. Test-time control follows Algorithm 1 for ε-ES and Algorithm 3 for the first-order variants; the Gaussian-refit variants are Algorithms 4 and 5. The steering horizon is one chunk ($H = 16$) everywhere, and $\lambda = 1$ except on `pointmaze-giant`, where $\lambda = 0.6$. ε-ES uses

Algorithm 2: VPFLOW training

Input: Offline dataset $\mathcal{D}$; horizon H; actor period M; iterations T; step sizes η_ϕ, η_θ
Output: trained parameters ϕ and θ

initialize $\phi = (E_\phi, f_\phi, V_{\phi,1}, V_{\phi,2})$ and $\theta = (v_\theta, \pi_\theta)$;
for $k = 1, \dots, T$ **do**
  sample $(s_t, a_{t:t+H-1}, s_{t+1:t+H}) \sim \mathcal{D}$; sample g^V (mixture) and g^π (on-trajectory);
  $R_{t+h} \leftarrow (1-\gamma^{d_h})/(\gamma-1)$ for $h = 0, \dots, H$;
  $M_{t+h} \leftarrow \mathbb{1}[g^V$ is $d_h \geq 0$ steps ahead of $s_{t+h}]$;
  $z_t \leftarrow E_\phi(s_t)$, $z_{g^V} \leftarrow E_\phi(g^V)$;
  $\hat{z}_{t+h} \leftarrow f_\phi(z_t, a_{t:t+h-1})$ for $h = 1, \dots, H$; // Equation 11
  $\phi \leftarrow \phi - \eta_\phi \nabla_\phi \mathcal{L}_{\text{VP}}(\phi)$; // Equation 6; no target network
  if $k \bmod M = 0$ **then** // policy improvement with ϕ frozen
    $\varepsilon \sim \mathcal{N}(0, I)$, $\boldsymbol{a} \leftarrow \pi_\theta(z_t, E_\phi(g^\pi), \varepsilon)$;
    $\tilde{z}_{t+H} \leftarrow f_\phi(z_t, \boldsymbol{a})$;
    $q_\theta \leftarrow \gamma^H \min_j V_{\phi,j}(\tilde{z}_{t+H}, E_\phi(g^\pi))$;
    $\theta \leftarrow \theta - \eta_\theta \nabla_\theta \mathcal{L}_\pi(\theta)$; // Equation 8

$N = 512$ candidates and $E = 64$ elites throughout, so a K-iteration run costs $512K$ decoded rollouts; ε-Adam and ε-SGD carry a single candidate and cost K forward–backward passes. Per-family values of the steering optimizer, K, $\sigma_{\max}$, $\sigma_{\min}$, ρ, η, β, and H_{exec} are listed in Table 4. The controller queries only the learned model; no environment simulator is used at test time.

Evaluation. Each dataset is evaluated on its five official tasks with 25 episodes per task in vectorized environments, and success is the official OGBench indicator that the goal is reached within the episode horizon. We report the mean and standard deviation across eight training seeds (four for pixel tasks), matching the official OGBench protocol (Park et al., 2025a), and paired bootstrap confidence intervals for controller comparisons that share a checkpoint.

B.7 PLANNER VARIANTS

The planners compared in Table 2 and Appendix E.2 share the decoder Dec of Algorithm 1 and the objective A of Equation 10; they differ in the search variable and in the update. ε-CEM (Algorithm 4) refits one diagonal Gaussian to the elite set at every generation instead of carrying the elites themselves, so a badly placed mean can move the whole population away from the best candidate seen. $\mathcal{A}$-CEM (Algorithm 5) applies the same refit to raw action sequences, warm-started at the policy mode. ε-SGD and ε-Adam (Algorithm 3) carry a single candidate and ascend A directly, spending K forward–backward passes against the KN forward passes of the population methods. All variants share the $K{=}0$ fallback to the policy chunk, and reported results use the deterministic execution path; training-time rollouts add small Gaussian exploration noise to the executed plan.

Design notes on ε-ES. The search is gradient-free: ε-ES queries advantage A only through forward evaluations, so it applies unchanged when the objective is flat or discontinuous in $\boldsymbol{\epsilon}$, as on the manipulation tasks of Section 5.3. Elitist replacement makes the best elite score monotone in the generation, and because the elites start at the policy mode, every elite at every generation scores at least as high as the reactive policy's chunk under the model. The final selection samples an elite from a softmax over scores with temperature β instead of taking the argmax, which hedges among near-tied plans whose ordering the model cannot certify; since all elites clear the policy mode, this sampling cannot regress below it either. Setting $K = 0$ returns the policy chunk directly, the no-steering control used throughout Section 5, and the controller otherwise executes the first H_{exec} actions of the selected plan and re-steers at the state reached.

C DERIVATIONS

C.1 CLOSED-FORM MONTE CARLO TARGETS

Fix a trajectory in which a future state at index $t + d$ is relabeled as the goal, and assume the goal is first satisfied at that index.

Algorithm 3: ε-SGD and ε-Adam: first-order steering in noise space

Input: state s_t, goal g; steps K; step size η; decay ρ; execution length H_{exec}
Output: the next H_{exec} actions to execute

$z_t \leftarrow E_\phi(s_t),\ z_g \leftarrow E_\phi(g)$;
if $K = 0$ **then return** the first H_{exec} actions of $\text{Dec}(\mathbf{0})$ `// policy chunk, no steering;`
$\boldsymbol{\varepsilon}^{(0)} \leftarrow \mathbf{0}; m_0, v_0 \leftarrow \mathbf{0}$; `// one candidate, no population`
for $k = 1, \dots, K$ **do**
  $\eta_k \leftarrow \eta \rho^{k-1}$;
  $g^{(k)} \leftarrow \nabla_{\boldsymbol{\varepsilon}} A(\text{Dec}(\boldsymbol{\varepsilon}^{(k-1)}); z_t, z_g)$; `// Equation 18`
  if *ε-SGD* **then**
    $\boldsymbol{\varepsilon}^{(k)} \leftarrow \boldsymbol{\varepsilon}^{(k-1)} + \eta_k\, g^{(k)}$;
  else
    $m_k \leftarrow 0.9 m_{k-1} + 0.1 g^{(k)},\ v_k \leftarrow 0.999 v_{k-1} + 0.001 (g^{(k)})^2$;
    $\boldsymbol{\varepsilon}^{(k)} \leftarrow \boldsymbol{\varepsilon}^{(k-1)} + \eta_k\, \hat{m}_k / (\sqrt{\hat{v}_k} + 10^{-8})$ with $\hat{m}_k = m_k/(1-0.9^k)$, $\hat{v}_k = v_k/(1-0.999^k)$;
return the first H_{exec} actions of $\text{Dec}(\boldsymbol{\varepsilon}^{(K)})$;

Algorithm 4: ε-CEM: Gaussian refit in noise space

Input: state s_t, goal g; population N; elites E; generations K; scales $\sigma_{\max}, \sigma_{\min}$; decay ρ; temperature β; execution length H_{exec}
Output: the next H_{exec} actions to execute

$z_t \leftarrow E_\phi(s_t),\ z_g \leftarrow E_\phi(g)$;
if $K = 0$ **then return** the first H_{exec} actions of $\text{Dec}(\mathbf{0})$ `// policy chunk, no steering;`
$\mu \leftarrow \mathbf{0}, \sigma \leftarrow \mathbf{1}$; `// one diagonal Gaussian over` $\boldsymbol{\varepsilon}$
for $k = 1, \dots, K$ **do**
  $\boldsymbol{\varepsilon}^{(i)} \sim \mathcal{N}(\mu, \sigma^2), i = 1, \dots, N$;
  $A_i \leftarrow A(\text{Dec}(\boldsymbol{\varepsilon}^{(i)}); z_t, z_g)$; `// Equation 10`
  $\mathcal{E} \leftarrow$ top-E of $\{\boldsymbol{\varepsilon}^{(i)}\}$ by A_i; $w_e \propto \exp(\beta A_e)$ over $\mathcal{E}$;
  $\mu \leftarrow \sum_e w_e\, \boldsymbol{\varepsilon}^{(e)},\ \sigma \leftarrow \big(\sum_e w_e (\boldsymbol{\varepsilon}^{(e)} - \mu)^2\big)^{1/2}$; `// softmax refit`
  $\sigma \leftarrow \text{clip}(\sigma, \sigma_{\min}, \max(\sigma_{\min}, \sigma_{\max} \rho^{k-1}))$;
$e^\star \sim p(e) \propto \exp(\beta A_e)$;
return the first H_{exec} actions of $\text{Dec}(\boldsymbol{\varepsilon}^{(e^\star)})$;

Reward convention. Equation 1 rewards 1 on arrival and 0 elsewhere, so the discounted return from the current index is γ^d. Our implementation uses the shifted, equivalent form $r(s_t, g) - 1 \in \{-1, 0\}$, which charges one unit for every step taken before arrival, as in OGBench (Park et al., 2025a). The two are related by the increasing affine map $u \mapsto (u-1)/(1-\gamma)$ and therefore rank candidate plans identically; we implement the shifted form because its targets spread over $[(\gamma-1)^{-1}, 0]$ instead of compressing toward zero at large d, which keeps the regression well conditioned on long-horizon tasks.

The target. Under the shifted form the per-step rewards are -1 for d steps and 0 upon satisfaction, so

$$R(d) = \sum_{k=0}^{d-1} \gamma^k (-1) = (-1)\,\frac{1-\gamma^d}{1-\gamma} = \frac{1-\gamma^d}{\gamma-1}, \tag{15}$$

Equation 15 is exactly Equation 3. Random-dataset goals that do not appear on the current trajectory are masked out of $\mathcal{L}_{\text{VP}}$; they do not receive a fabricated distance.

These targets are functions of indices in $\mathcal{D}$. They do not depend on f_ϕ or on a bootstrapped $\bar{V}$. That is why default VPFLOW training does not need a target encoder: the regression problem is supervised.

C.2 GAE RANKING AND N-STEP RETURNS

The H-step latent return that a standard latent planner maximizes is

$$G_H(\boldsymbol{a}; z_t, z_g) = \sum_{h=0}^{H-1} \gamma^h r_{t+h} + \gamma^H V_\phi(\hat{z}_{t+H}, z_g), \quad \hat{z}_{t+h+1} = f_\phi(\hat{z}_{t+h}, a_{t+h}), \ \ \hat{z}_t := z_t. \tag{16}$$

Algorithm 5: $\mathcal{A}$-CEM: Gaussian refit in action space

Input: state s_t, goal g; population N; elites E; iterations K; scales $\sigma_{\max}, \sigma_{\min}$; decay ρ; temperature β; execution length H_{exec}

Output: the next H_{exec} actions to execute

$z_t \leftarrow E_\phi(s_t),\ z_g \leftarrow E_\phi(g)$;

if $K = 0$ **then return** the first H_{exec} actions of $\text{Dec}(\mathbf{0})$ `// policy chunk, no steering;`

$\mu_a \leftarrow \text{Dec}(\mathbf{0}), \sigma_a \leftarrow \sigma_{\max}$; `// warm start at the policy mode`

for $k = 1, \ldots, K$ **do**

  $\sigma_a \leftarrow \text{clip}(\sigma_a, \sigma_{\min}, \max(\sigma_{\min}, \sigma_{\max}\rho^{k-1}))$;

  $\boldsymbol{a}^{(i)} \sim \mathcal{N}(\mu_a, \sigma_a^2)$ clipped to $[-1, 1], i = 1, \ldots, N$;

  $A_i \leftarrow A(\boldsymbol{a}^{(i)}; z_t, z_g)$; `// Equation 10 on raw action sequences`

  $\mathcal{E} \leftarrow$ top-E of $\{\boldsymbol{a}^{(i)}\}$ by A_i; $w_e \propto \exp(\beta A_e)$ over $\mathcal{E}$;

  $\mu_a \leftarrow \sum_e w_e \boldsymbol{a}^{(e)}, \sigma_a \leftarrow \left(\sum_e w_e(\boldsymbol{a}^{(e)} - \mu_a)^2\right)^{1/2}$;

return the first H_{exec} actions of μ_a;

Let $V_{t+h} = V_\phi(\hat{z}_{t+h}, z_g)$, $r_{t+h} = -1$, and $\delta_{t+h} = r_{t+h} + \gamma V_{t+h+1} - V_{t+h}$ as in Equation 9. The planning objective of Equation 10 is $A_t = \sum_{h=0}^{H-1} (\gamma\lambda)^h \delta_{t+h}$. For $\lambda = 1$,

$$A_t = \sum_{h=0}^{H-1} \gamma^h (r_{t+h} + \gamma V_{t+h+1} - V_{t+h}) = \sum_{h=0}^{H-1} \gamma^h r_{t+h} + \gamma^H V_{t+H} - V_t, \tag{17}$$

The telescoping sum in Equation 17 shows that A_t equals the H-step return of Equation 16 up to the trajectory-independent shift $-V_t$. Both steering families therefore rank candidates by n-step value at $\lambda = 1$. For $\lambda < 1$ the same expansion is a geometrically weighted advantage, which downweights value differences asserted deep inside an imagined rollout at the cost of myopia. We keep $r_{t+h} = -1$ on imagined steps because the method has no learned reward model and does not assume that the goal is reached inside H; any remaining progress is the job of V_{t+H}.

C.3 Noise-space decoding

The distilled one-step policy is a deterministic map $\boldsymbol{a} = \pi_\theta(z, z_g, \varepsilon)$ with $\varepsilon \sim \mathcal{N}(0, I)$ at training time. Setting $\varepsilon = 0$ recovers the actor mode used in the no-steering control. Perturbing ε traces a neighborhood of that mode in the coordinates of CFM, rather than in raw action space. Closed-loop chunk decoding uses the same scan as training imagination: each decoded chunk is applied through f_ϕ, then the next ε_c is decoded from the resulting latent. This keeps the planner's action-chunk length equal to the training chunk length, which is required for Transformer dynamics whose context was trained at width H.

C.4 First-order steering

ε-SGD and ε-Adam (Algorithm 3) maximize A in Equation 10 by ascent on $\boldsymbol{\varepsilon}$, starting from $\boldsymbol{\varepsilon}^{(0)} = \mathbf{0}$; K counts actual ascent steps, and the step size is annealed as $\eta_k = \eta\rho^{k-1}$ so that $K = 1$ is unchanged by ρ. ε-SGD uses the raw gradient of Equation 18. ε-Adam maintains $m_k = 0.9m_{k-1} + 0.1g^{(k)}$ and $v_k = 0.999v_{k-1} + 0.001(g^{(k)})^2$ with the usual bias corrections $\hat{m}_k = m_k/(1 - 0.9^k)$, $\hat{v}_k = v_k/(1 - 0.999^k)$, and takes $\boldsymbol{\varepsilon}^{(k)} = \boldsymbol{\varepsilon}^{(k-1)} + \eta_k \hat{m}_k/(\sqrt{\hat{v}_k} + 10^{-8})$ (Kingma & Ba, 2015). The per-coordinate displacement is then close to η_k irrespective of the gradient magnitude, which is why the useful range of η differs by an order of magnitude between the two variants (Appendix D). We do not clip ε; decoded actions are clipped to $[-1, 1]$. The same optimizers run in raw action space for the action-space controls of Section 5.3, in which case the iterate is clipped at every step. Because a single candidate is carried, first-order steering costs K forward–backward passes against the KN forward passes of ε-ES; the two are equated by world-model evaluations rather than by K in Appendix D.

D HYPERPARAMETERS AND COMPUTE

D.1 TRAINING

Table 3 lists the training configuration, which is fixed within each observation modality and shared by every dataset in Tables 5 and 6.

Table 3: Training hyperparameters.

Quantity	**State**	**Pixel**
Gradient updates	1,000,000	500,000
Batch size	256	256
Optimizer	Adam	Adam
Learning rate (cosine decay)	2×10^{-4}	2×10^{-4}
Gradient-norm clip	10	10
Discount γ	0.99	0.99
Training unroll and action chunk H	16	16
Actor update period M	1	1
Value heads	2	2
Value-head MLP	3×512, LayerNorm, GELU	3×512, LayerNorm, GELU
Dynamics (cross-attention)	192 / 3 layers / 3 heads	192 / 3 layers / 3 heads
Dynamics dropout	0.1	0.1
Flow networks	192 / 3 layers / 3 heads	192 / 3 layers / 3 heads
Action-token attention	bidirectional	prefix-LM
Flow integration steps (teacher)	10	10
α_{CFM}	1	1
α_{distill}	200	50
Encoder	identity	IMPALA-Small
Random-crop probability	–	0.5
Value-goal mixture (current/future/random)	0.2/0.5/0.3	0.2/0.5/0.3

D.2 STEERING

Table 4 lists the steering settings used for the reported results. The horizon is one chunk ($H = 16$) everywhere, ε-ES always uses $N = 512$ candidates and $E = 64$ elites, and the softmax temperature is $\beta = 0.1$ on navigation and 0.5 on manipulation. The step-size decay is $\rho = 0.5$ where listed and $\rho = 1$ (constant) otherwise; $K = 1$ is unaffected by ρ in either family.

Table 4: Steering settings per environment family. $\sigma_{\max}$ and $\sigma_{\min}$ apply to ε-ES; η is the step size of the first-order variants.

Family	Optimizer	K	$\sigma_{\max}$	$\sigma_{\min}$	η	ρ	λ	H_{exec}
PointMaze (state)	ε-ES	4	1.0	0.05	–	1	0.6	16
AntMaze (state)	ε-SGD	4–5	–	–	0.05–0.2	1	1	3
Cube, Scene, Puzzle (state)	ε-ES	2–6	0.3–1.0	0	–	1	1	3
visual-AntMaze	ε-Adam	2–5	–	–	0.03–0.1	1	1	1–3
visual-Cube, Scene, Puzzle	ε-ES	1–5	0.5–1.0	0.05	–	0.5	1	3–8

Search grid. The steering settings in Table 4, used for the results of Section 5.3, were selected by sweeping $K \in \{0, \dots, 10\}$, $\sigma_{\max} \in \{0.1, 0.2, 0.3, 0.5, 0.8, 1.0, 2.0\}$, $\sigma_{\min} \in \{0, 0.05, 0.1, 0.2, 1.0\}$, $\beta \in \{0.05, 0.1, 0.3, 0.5, 0.9\}$, $\rho \in \{0.5, 0.6, 0.75, 1.0\}$, $H_{\text{exec}} \in \{1, 2, 3, 5, 8, 16\}$, and $\lambda \in \{0, 0.1, 0.3, 0.5, 0.6, 0.65, 0.7, 0.75, 0.85, 0.95, 1\}$. The useful range of η differs between the first-order variants because Adam normalizes the per-coordinate step: $\eta \in \{0.02, \dots, 1.0\}$ for ε-SGD and $\eta \in \{0.01, \dots, 0.2\}$ for ε-Adam. Selection uses exploratory evaluation seeds and is frozen before the reported runs.

Table 5: Success rate (%) on OGBench **state-based** datasets. Mean ± std across 8 seeds; baseline numbers from Park et al. (2025a); best per row in bold.

Env.	Type	Dataset	GCBC	GCIVL	GCIQL	QRL	CRL	VPFLOW
PointMaze	Nav.	pointmaze-medium-navigate-v0	$9_{\pm 6}$	$63_{\pm 6}$	$53_{\pm 8}$	$82_{\pm 8}$	$29_{\pm 7}$	$\mathbf{98_{\pm 2}}$
		pointmaze-large-navigate-v0	$29_{\pm 6}$	$45_{\pm 5}$	$34_{\pm 3}$	$86_{\pm 9}$	$39_{\pm 7}$	$\mathbf{99_{\pm 1}}$
		pointmaze-giant-navigate-v0	$1_{\pm 2}$	$0_{\pm 0}$	$0_{\pm 0}$	$68_{\pm 7}$	$27_{\pm 10}$	$\mathbf{90_{\pm 3}}$
		pointmaze-teleport-navigate-v0	$25_{\pm 3}$	$45_{\pm 3}$	$24_{\pm 7}$	$4_{\pm 4}$	$24_{\pm 6}$	$\mathbf{64_{\pm 2}}$
AntMaze	Nav.	antmaze-medium-navigate-v0	$29_{\pm 4}$	$72_{\pm 8}$	$71_{\pm 4}$	$88_{\pm 3}$	$95_{\pm 1}$	$\mathbf{96_{\pm 1}}$
		antmaze-large-navigate-v0	$24_{\pm 2}$	$16_{\pm 5}$	$34_{\pm 4}$	$75_{\pm 6}$	$83_{\pm 4}$	$\mathbf{85_{\pm 2}}$
		antmaze-giant-navigate-v0	$0_{\pm 0}$	$0_{\pm 0}$	$0_{\pm 0}$	$14_{\pm 3}$	$16_{\pm 3}$	$\mathbf{33_{\pm 3}}$
		antmaze-teleport-navigate-v0	$26_{\pm 3}$	$39_{\pm 3}$	$35_{\pm 5}$	$35_{\pm 5}$	$53_{\pm 2}$	$\mathbf{58_{\pm 3}}$
Cube	Play	cube-single-play-v0	$6_{\pm 2}$	$53_{\pm 4}$	$68_{\pm 6}$	$5_{\pm 1}$	$19_{\pm 2}$	$\mathbf{98_{\pm 1}}$
		cube-double-play-v0	$1_{\pm 1}$	$36_{\pm 3}$	$40_{\pm 5}$	$1_{\pm 0}$	$10_{\pm 2}$	$\mathbf{60_{\pm 1}}$
		cube-triple-play-v0	$1_{\pm 1}$	$1_{\pm 0}$	$3_{\pm 1}$	$0_{\pm 0}$	$4_{\pm 1}$	$\mathbf{10_{\pm 2}}$
		cube-quadruple-play-v0	$0_{\pm 0}$	$0_{\pm 0}$	$0_{\pm 0}$	$0_{\pm 0}$	$0_{\pm 0}$	$\mathbf{1_{\pm 0}}$
Scene	Play	scene-play-v0	$5_{\pm 1}$	$42_{\pm 4}$	$51_{\pm 4}$	$5_{\pm 1}$	$19_{\pm 2}$	$\mathbf{78_{\pm 2}}$
Puzzle	Play	puzzle-3x3-play-v0	$2_{\pm 0}$	$6_{\pm 1}$	$\mathbf{95_{\pm 1}}$	$1_{\pm 0}$	$3_{\pm 1}$	$93_{\pm 1}$
		puzzle-4x4-play-v0	$0_{\pm 0}$	$13_{\pm 2}$	$26_{\pm 3}$	$0_{\pm 0}$	$0_{\pm 0}$	$\mathbf{79_{\pm 4}}$
		puzzle-4x5-play-v0	$0_{\pm 0}$	$7_{\pm 1}$	$14_{\pm 1}$	$0_{\pm 0}$	$1_{\pm 0}$	$\mathbf{20_{\pm 2}}$
		puzzle-4x6-play-v0	$0_{\pm 0}$	$10_{\pm 2}$	$\mathbf{12_{\pm 1}}$	$0_{\pm 0}$	$4_{\pm 1}$	$\mathbf{12_{\pm 1}}$

Table 6: Success rate (%) on OGBench **pixel-based** datasets. Mean ± std across 4 seeds; baseline numbers from Park et al. (2025a); best per row in bold.

Env.	Type	Dataset	GCBC	GCIVL	GCIQL	QRL	CRL	VPFLOW
AntMaze	Nav.	visual-antmaze-medium-navigate-v0	$11_{\pm 2}$	$22_{\pm 2}$	$11_{\pm 1}$	$0_{\pm 0}$	$94_{\pm 1}$	$\mathbf{96_{\pm 1}}$
		visual-antmaze-large-navigate-v0	$4_{\pm 0}$	$5_{\pm 1}$	$4_{\pm 1}$	$0_{\pm 0}$	$\mathbf{84_{\pm 1}}$	$\mathbf{84_{\pm 1}}$
		visual-antmaze-giant-navigate-v0	$0_{\pm 0}$	$1_{\pm 1}$	$0_{\pm 0}$	$0_{\pm 0}$	$\mathbf{47_{\pm 2}}$	$10_{\pm 1}$
		visual-antmaze-teleport-navigate-v0	$5_{\pm 1}$	$8_{\pm 1}$	$6_{\pm 1}$	$6_{\pm 3}$	$48_{\pm 2}$	$\mathbf{54_{\pm 2}}$
Cube	Play	visual-cube-single-play-v0	$5_{\pm 1}$	$60_{\pm 5}$	$30_{\pm 5}$	$41_{\pm 15}$	$31_{\pm 15}$	$\mathbf{83_{\pm 2}}$
		visual-cube-double-play-v0	$1_{\pm 1}$	$10_{\pm 2}$	$1_{\pm 1}$	$5_{\pm 0}$	$2_{\pm 1}$	$\mathbf{30_{\pm 1}}$
		visual-cube-triple-play-v0	$15_{\pm 2}$	$14_{\pm 2}$	$15_{\pm 1}$	$16_{\pm 1}$	$17_{\pm 2}$	$\mathbf{20_{\pm 1}}$
		visual-cube-quadruple-play-v0	$\mathbf{8_{\pm 1}}$	$0_{\pm 0}$	$7_{\pm 1}$	$5_{\pm 1}$	$4_{\pm 1}$	$6_{\pm 0}$
Scene	Play	visual-scene-play-v0	$12_{\pm 2}$	$25_{\pm 3}$	$12_{\pm 2}$	$10_{\pm 1}$	$11_{\pm 2}$	$\mathbf{33_{\pm 2}}$
Puzzle	Play	visual-puzzle-3x3-play-v0	$0_{\pm 0}$	$\mathbf{21_{\pm 1}}$	$1_{\pm 2}$	$1_{\pm 1}$	$0_{\pm 0}$	$20_{\pm 0}$
		visual-puzzle-4x4-play-v0	$10_{\pm 1}$	$\mathbf{60_{\pm 5}}$	$16_{\pm 4}$	$0_{\pm 0}$	$10_{\pm 6}$	$26_{\pm 2}$
		visual-puzzle-4x5-play-v0	$5_{\pm 2}$	$17_{\pm 1}$	$7_{\pm 2}$	$0_{\pm 0}$	$6_{\pm 1}$	$\mathbf{20_{\pm 0}}$
		visual-puzzle-4x6-play-v0	$2_{\pm 1}$	$\mathbf{15_{\pm 1}}$	$2_{\pm 1}$	$0_{\pm 0}$	$3_{\pm 1}$	$7_{\pm 0}$

D.3 SOFTWARE

All experiments ran on NVIDIA A100 GPUs. The implementation is in JAX (CUDA 12) with Equinox and Flax, on the official OGBench datasets and evaluation protocol (Park et al., 2025a).

E EXTENDED QUANTITATIVE RESULTS

E.1 PER-DATASET OGBENCH TABLES

Tables 5 and 6 report the per-dataset results summarized in Table 1. Baseline entries are the official OGBench numbers (Park et al., 2025a); the hierarchical HIQL baseline is omitted because VPFLOW is a flat controller. Values are mean ± standard deviation across training seeds; the best entry in each row is in bold (ties are both bolded). VPFLOW uses Algorithm 1 with the steering settings of Table 4.

E.2 CHOOSING THE STEERING OPTIMIZER

The noise space of Section 4.5 is a set of coordinates, and several optimizers can run over it. Besides ε-ES, the advantage objective A is differentiable in $\boldsymbol{\varepsilon}$ because π_θ, f_ϕ, and V_ϕ all are, so we can ascend it directly from the same origin,

$$\boldsymbol{\varepsilon}^{(k)} = \boldsymbol{\varepsilon}^{(k-1)} + \eta_k \, \nabla_{\boldsymbol{\varepsilon}} A\big(\mathrm{Dec}(\boldsymbol{\varepsilon}^{(k-1)}); z_t, z_g\big), \qquad \boldsymbol{\varepsilon}^{(0)} = \mathbf{0}, \qquad \eta_k = \eta \rho^{\,k-1}, \tag{18}$$

as ε-SGD, or with the bias-corrected Adam direction (Kingma & Ba, 2015) as ε-Adam, whose per-coordinate step stays near η_k regardless of gradient scale. The two families read one objective and pay differently. ε-ES spends KN forward passes on a population and needs no smoothness; first-order steering spends K forward–backward passes on one candidate and needs the landscape to be locally informative.

Table 7 compares the two task types showing the results of different planner using the best K at each dataset.

Table 7: **Success rate (%, ↑) demonstrating the effectiveness of noise steering and optimizer choices on OGBench datasets.** In general, first-order method is more effective on navigation tasks, while zeroth-order method is better on manipulation tasks. Noise steering with value function allows the policy to improve itself at test time using imagined rollouts.

Task Type	Dataset	VPFLOW (ε-ES)	VPFLOW (ε-SGD or ε-Adam)	VPFLOW w/o steering ($K = 0$)
Navigation	pointmaze-medium-navigate-v0	98 ± 1	$\mathbf{98 \pm 2}$	97 ± 1
	pointmaze-large-navigate-v0	98 ± 1	$\mathbf{99 \pm 1}$	97 ± 1
	pointmaze-giant-navigate-v0	77 ± 3	$\mathbf{90 \pm 3}$	54 ± 3
	pointmaze-teleport-navigate-v0	60 ± 3	$\mathbf{64 \pm 2}$	57 ± 2
	antmaze-medium-navigate-v0	90 ± 2	$\mathbf{96 \pm 1}$	83 ± 4
	antmaze-large-navigate-v0	75 ± 3	$\mathbf{85 \pm 2}$	70 ± 3
	antmaze-giant-navigate-v0	26 ± 2	$\mathbf{33 \pm 3}$	12 ± 1
	antmaze-teleport-navigate-v0	44 ± 5	$\mathbf{58 \pm 3}$	34 ± 2
	visual-antmaze-medium-navigate-v0	90 ± 2	$\mathbf{96 \pm 1}$	84 ± 1
	visual-antmaze-large-navigate-v0	70 ± 5	$\mathbf{84 \pm 1}$	64 ± 4
	visual-antmaze-giant-navigate-v0	0 ± 0	$\mathbf{10 \pm 1}$	5 ± 2
	visual-antmaze-teleport-navigate-v0	40 ± 4	$\mathbf{54 \pm 2}$	33 ± 3
Manipulation	cube-single-play-v0	93 ± 1	$\mathbf{98 \pm 1}$	26 ± 2
	cube-double-play-v0	$\mathbf{60 \pm 1}$	55 ± 2	2 ± 1
	cube-triple-play-v0	$\mathbf{10 \pm 2}$	5 ± 1	0 ± 0
	cube-quadruple-play-v0	$\mathbf{1 \pm 0}$	0 ± 0	0 ± 0
	scene-play-v0	$\mathbf{78 \pm 2}$	64 ± 2	25 ± 2
	puzzle-3x3-play-v0	$\mathbf{93 \pm 1}$	64 ± 5	1 ± 1
	puzzle-4x4-play-v0	$\mathbf{79 \pm 4}$	57 ± 4	1 ± 1
	puzzle-4x5-play-v0	$\mathbf{20 \pm 2}$	0 ± 0	0 ± 0
	puzzle-4x6-play-v0	$\mathbf{12 \pm 1}$	0 ± 0	0 ± 0
	visual-cube-single-play-v0	$\mathbf{83 \pm 2}$	67 ± 1	56 ± 3
	visual-cube-double-play-v0	$\mathbf{30 \pm 1}$	27 ± 3	16 ± 2
	visual-cube-triple-play-v0	$\mathbf{20 \pm 1}$	17 ± 2	14 ± 2
	visual-cube-quadruple-play-v0	$\mathbf{6 \pm 0}$	3 ± 1	3 ± 2
	visual-scene-play-v0	$\mathbf{33 \pm 2}$	11 ± 2	6 ± 1
	visual-puzzle-3x3-play-v0	$\mathbf{20 \pm 0}$	0 ± 0	0 ± 0
	visual-puzzle-4x4-play-v0	$\mathbf{26 \pm 2}$	0 ± 0	0 ± 0
	visual-puzzle-4x5-play-v0	$\mathbf{20 \pm 0}$	0 ± 0	0 ± 0
	visual-puzzle-4x6-play-v0	$\mathbf{7 \pm 0}$	0 ± 0	0 ± 0

F Discussion and Limitations

Scope. VPFLOW is an offline goal-conditioned controller. It does not claim a general world model for arbitrary dense rewards, nor a pixel reconstruction model. The method assumes that hindsight temporal distances in $\mathcal{D}$ are a useful proxy for test-time goal reaching, which is the standard OGBench contract but fails when the offline trajectories cannot be stitched from the test start to the test goal. The closed-form MC target is likewise specific to sparse goal-reaching rewards; dense or stochastic rewards would require a learned reward model or TD bootstrapping.

Where the pixel results lag. Table 6 shows that VPFLOW trails CRL on `visual-antmaze-giant` and GCIVL on the larger visual puzzles. These are the longest-horizon pixel tasks in their families, and both require the encoder to recover fine-grained state (position in a large maze, the configuration of many buttons) from a single frame. Because steering ranks candidates only through V_ϕ, an encoder that cannot resolve this state yields a flat value landscape that no amount of search can exploit. We therefore regard visual value estimation, not the search coordinate, as the current bottleneck.

Over-optimization of the model. Steering can exploit errors in f_ϕ, and Section 5.3 reports the cases where it does: several iteration curves peak and then decline even though the imagined score that ε-ES optimizes is monotone by construction. We report these curves rather than their envelope. Uncertainty-aware scoring, in the spirit of the pessimistic penalties used in offline model-based RL (Yu et al., 2020; Kidambi et al., 2020), is a natural extension.

Choosing the steering optimizer. Section 5.3 reports a consistent split between locomotion and manipulation and offers a reading of it in terms of the shape of $S(\varepsilon)$, but we select the optimizer per environment family on exploratory seeds rather than deriving it. A practitioner facing a new domain therefore has a rule of thumb, not a test. A cheap diagnostic, for instance the ratio between the gradient norm and the score spread of a small mutation population at the first re-steering step, would turn the observation into a procedure, and we have not built one.

Design choices that are not contributions. Twin value heads with a minimum are the standard overestimation control from clipped double Q-learning (Fujimoto et al., 2019). The flow actor, one-step distillation, and action chunks are adopted from FQL (Park et al., 2025b) and MAC (Park et al., 2026a); our contribution is the value-predictive model that scores them and the noise-space steering that refines them. Neither ε-ES nor Adam is new as an optimizer; what we claim is the coordinate system they run in and the evidence about which of them suits which task family. We do not prove that value consistency implies feasible dynamics; the objective sweep of Section 5.2 and the PointMaze rollout overlays are the empirical substitute.

Hyperparameters. Training hyperparameters are shared within each observation modality, but steering settings differ across environment families (Table 4). This is less convenient than a single global configuration, and we state it explicitly.