Learning asymmetric Laplacian representations in Hilbert space

Anonymous authors
Paper under double-blind review

Abstract

Laplacian representation learning is a highly promising generally and in reinforcement learning, where an important challenge is to retain the asymmetries of directed and stochastic dynamics in the environment. In the self-adjoint setting, prior methods use ordered orthogonality and a smoothness objective to learn a set of spectral representations corresponding to lowest-magnitude eigenvalues. Such ordering dynamics does not extend to the asymmetric Laplacian. Unlike a self-adjoint Laplacian, an asymmetric Laplacian can have complex eigenvalues and non-orthogonal complex eigenfunctions; that is, its right and adjoint-left eigenfunctions can differ and form bi-orthogonality. These properties can complicate approaches that attempt to learn an ordered eigenbasis. One might therefore expect an asymmetric solution to require a substantially more complicated approach. We instead keep the representation learning problem to a similar complexity of prior symmetric approaches and find that, by targeting an unordered subspace, we can learn vectors that approximate mixtures of the asymmetric Laplacian eigenvectors we seek. Careful unmixing then yields good approximations of low magnitude asymmetric eigenfunctions. We compare the obtained asymmetric eigenfunctions with oracle solutions using cosine similarity across multiple domains. Additionally, we show that the learnt eigenfunctions can be used in domains to obtain hitting time estimates and compare them to the prior symmetric Laplacian based commute-time distance, showing improved distance learning in directed and stochastic settings.

1 Introduction

Laplacian representations have proved useful for reinforcement learning in continuous and high-dimensional settings. A recent approach, ALLO (Gomez et al., 2024), is able to obtain good approximations of eigenvectors associated with the lowest-magnitude eigenvalues; however, it assumes that the Laplacian is symmetric. Prior approaches, Generalized Graph Drawing Objectives (GGDO) (Wang et al., 2021) and Graph Drawing Objectives (GDO) (Wu et al., 2019), also make this symmetry assumption. Subsequent works have utilised these representations methods to approximate commute-time distances (Wang et al., 2023a), discover bottlenecks (Liang et al., 2026) and form spectral clusters for decision-time planning (Shehmar et al., 2026). Furthermore, recent approaches for exploration such as Deep Covering Eigenoptions (DCEO) (Klissarov & Machado, 2023) and Hierarchical Novel Eigen Options (HNEO) (Vijayan et al., 2026) also have to assume in representation that the Laplacian is symmetric. But forcing symmetry in directed or stochastic domains can miss structural information about an environment's asymmetric properties. Therein, learning representations that retain these properties remains an important challenge.

Whilst there is clear motivation for learning a spectrum of asymmetric Laplacian operators, the difficulty of learning complex eigenvectors and eigenvalues in continuous and high-dimensional settings via neural networks has been noted. Wang et al. (2023a, Section 5.1) explicitly notes the difficulty when discussing the challenge of learning asymmetric distances and asymmetric Laplacian representations that can scale to practical settings (such as robotic domains) as a "highly non-trivial challenge", further noting the difficulty in learning complex-valued eigenvalues. In this paper, we specifically focus on eigenvector and eigenvalue learning for the asymmetric case. For the forward Markov operator $\mathcal{P}$, formally defined in Equation 2, we learn real observation functions whose span is approximately invariant under that operator, and subsequently extract directed eigenfunctions through simply unmixing the solution. That is, the neural outputs need not themselves be eigenfunctions;

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.

unordered mixtures are sufficient. After obtaining an approximate eigenvector mixture, our problem then is simply: how do we unmix them to obtain our directed eigenvectors? Specifically, one learning phase is to learn a set of vectors F (real-valued feature functions) that span a subspace invariant approximately under $\mathcal{P}$, with the vectors potentially a mix of the desired asymmetric eigenvectors we want. And the second phase is to unmix those vectors toward eigenvector solutions. We call the proposed method asymmetric Laplacian representations (ALR).

In summary, ALR learns good approximations of eigenvectors corresponding to low-magnitude eigenvalues of the asymmetric Laplacian $\mathcal{L} = I - \mathcal{P}$ via neural network learning in continuous and high-dimensional settings. We present results in four grid-based and four multi-dimensional drone environments. We compare approximated asymmetric eigenfunctions obtained via ALR with oracle solutions and the current state of the art, ALLO (Gomez et al., 2024).

Our contributions are as follows: (1) We formulate asymmetric Laplacian representations (ALR), which approximate eigenvectors of asymmetric Laplacians associated with low magnitude eigenvalues in continuous and high-dimensional settings. We demonstrate state-of-the-art performance in both eigenvector and eigenvalue approximation accuracy compared with the related symmetric baseline, ALLO. (2) We introduce the learning of multi-horizon K_{γ_K} matrices to find coefficients to rotate subspace vectors into approximate Laplacian eigenvectors and analyze how performance varies as a function of γ_K. (3) We show that ALR provides an accurate approximation of the left zero-mode eigenvector, which corresponds to the stationary distribution. (4) We further show that the approximated eigenvectors and eigenvalues of an asymmetric Laplacian can be used to estimate hitting-time distances in continuous and high-dimensional settings.

2 Preliminaries

Markov decision process. A finite Markov decision process (MDP) is a five-tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \longrightarrow [0, 1]$ is the transition kernel, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, and $\gamma \in (0, 1)$ is the discount factor. Here $T(s, a, s')$ is the probability of the next state s' after taking action a in state s, with $\sum_{s' \in \mathcal{S}} T(s, a, s') = 1$. A fixed Markov policy π assigns a probability $\pi(a \mid s)$ to action a in state s and induces the one-step transition rule $P^\pi(s' \mid s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) T(s, a, s')$, where $\sum_{s' \in \mathcal{S}} P^\pi(s' \mid s) = 1$. Thus P^π is a row-stochastic transition matrix. For general state spaces, the same construction is expressed using a policy-induced Markov transition kernel $P^\pi(s, ds')$, with $P^\pi(s, S) = 1$. We use the Hilbert-space formulation in this section to describe both finite-state and continuous-state representations.

Hilbert Space and the asymmetric Laplacian. Let S be a state space, Σ a σ-algebra on S, ρ a probability measure on (S, Σ). Define

$$\mathcal{H} = L^2(S, \rho; \mathbb{C}), \qquad \langle f, g \rangle_\rho = \int_S \overline{f(u)} g(u)\, \rho(du), \qquad \|f\|_\rho^2 = \langle f, f \rangle_\rho. \tag{1}$$

Here $\mathbb{C}$ denotes the complex numbers, and the inner product $\langle f, g \rangle_\rho$ weights functions f and g by ρ. If $\langle f, g \rangle_\rho = 0$ then f and g are orthogonal. $L^2(S, \rho; \mathbb{C})$ consists of square-integrable complex-valued functions, identified up to equality ρ-almost everywhere. The forward transition kernel P^π induces the linear operator

$$(\mathcal{P} f)(u) = \int_S f(v)\, P^\pi(u, dv) = \mathbb{E}[f(Y) \mid X = u]. \tag{2}$$

Here X denotes the current state and Y denotes the next state under policy π. We take ρ to be the probability measure over current states induced by policy π, and assume that $\mathcal{P}$ is well defined and bounded on $\mathcal{H}$ (i.e. $\mathcal{P} : \mathcal{H} \to \mathcal{H}$ is a bounded linear operator). We assume that $P^\pi(u, \cdot)$ is absolutely continuous with respect to ρ. Then the Radon-Nikodym derivative $k(u, \cdot)$ gives the transition density

$$k(u, v) = \frac{dP^\pi(u, \cdot)}{d\rho}(v), \qquad (\mathcal{P} f)(u) = \int_S k(u, v) f(v)\, \rho(dv), \tag{3}$$

where $k(u, v) \geq 0$, and $\int_S k(u, v)\, \rho(dv) = 1$. For finite S with $\rho(v) > 0$, this becomes $k(u, v) = \frac{P^\pi(v|u)}{\rho(v)}$. The density does not have to be symmetric, that is, $k(u, v)$ need not equal $k(v, u)$, and thus

$\mathcal{P}$ does not have to satisfy the self-adjoint property $\mathcal{P} = \mathcal{P}^*$, where $*$ denotes the conjugate transpose. Let I denote the identity operator on $\mathcal{H}$. Then the asymmetric Laplacian is

$$\mathcal{L} = I - \mathcal{P}. \tag{4}$$

We denote the right eigenfunctions by $\phi_i^{\mathrm{r}} \in \Phi^{\mathrm{r}}$, the adjoint-left eigenfunctions by $\phi_i^{\mathrm{l}} \in \Phi^{\mathrm{l}}$, and the corresponding eigenvalues by $\lambda_i \in \Lambda$, where $\mathcal{L}\phi_i^{\mathrm{r}} = \lambda_i \phi_i^{\mathrm{r}}$ and $\mathcal{L}^*\phi_i^{\mathrm{l}} = \overline{\lambda_i}\phi_i^{\mathrm{l}}$. Note that, in the finite case, the stationary distribution μ satisfies $\mu^\top P^\pi = \mu^\top$, and $\mu^\top \mathcal{L} = 0$, where $\top$ denotes the transpose. Thus a stationary distribution is a left eigenvector associated with the zero eigenvalue of the asymmetric Laplacian $\phi_0^\top \mathcal{L} = 0$. For a right eigenfunction ϕ^r, expanding the eigenfunction equation gives $\mathcal{L}\phi^r = \lambda\phi \iff \mathcal{P}\phi^r = (1-\lambda)\phi^r$. In a finite state space, we then obtain the relation $\sum_j P_{ij}\phi^r(j) = (1-\lambda)\phi(i)$. When $\lambda \neq 1$, it can equivalently be written as

$$\phi(i) = \frac{1}{1-\lambda}\sum_j P_{ij}\phi^r(j) = \frac{1}{1-\lambda}\mathbb{E}[\phi^r(Y) \mid X = i]. \tag{5}$$

Thus an eigenfunction value is related to a probability-weighted average of its possible successor values. For a graph, these are outgoing neighbours; for a general state space, the sum is replaced by integration with $P^\pi(x, \mathrm{d}y)$.

Graph Drawing Objective. The graph drawing objective (GDO) from Wu et al. (2019) learns vectors from a subspace associated with the low-magnitude eigenvalues of a symmetric Laplacian in Hilbert space. The vectors are learnt as outputs of a neural network. We express these network output vectors as $F_\theta(x) = [f_1(x) \ \cdots \ f_m(x)] \in \mathbb{R}^{1\times m}$ where $f_\theta(x) = F_\theta(x)^\top \in \mathbb{R}^m$. Here θ denotes the trainable network parameters. Let n be the number of sampled state pairs. Define $M_F = \mathbb{E}_\rho[F_\theta(X)^\top F_\theta(X)]$. The GDO loss combines a smoothness term $\widehat{\mathcal{J}}_{\mathrm{smooth}}(\theta) = \frac{1}{2n}\sum_{i=1}^n \|F_\theta(X_i) - F_\theta(Y_i)\|_2^2$ with a weighted orthogonality and norm penalty $\mathcal{J}_{\mathrm{orth}}$:

$$\mathcal{J}_{\mathrm{orth}} = \|M_F - I_d\|_F^2 = \underbrace{\sum_{i\neq j}\langle f_i, f_j\rangle_\rho^2}_{\text{orthogonality term}} + \underbrace{\sum_i \left(\|f_i\|_\rho^2 - 1\right)^2}_{\text{norm term}}. \tag{6}$$

The off-diagonal terms discourage duplicate directions; the diagonal terms preserve the mean-square norm of each function, preventing the norm from collapsing to zero. The smoothness loss consists of the squared difference between $f(x)$ and $f(y)$, where both $f(x)$ and $f(y)$ contribute gradients to the network and therefore the update is completely symmetric. GGDO (Wang et al., 2021) improved upon GDO by adding hyperparameter weights to the orthogonality term; ALLO (Gomez et al., 2024) subsequently further improved upon GDDO by adding barriers, barrier coefficients, and stop-gradients for an asymmetric orthogonality loss, so it can learn the eigenvectors in order. In each approach, due to one-step sample pairs providing only very local propogation of values within the smoothness loss $\widehat{\mathcal{J}}_{\mathrm{smooth}}(\theta)$, a multi-step sampling approach is added. Specifically, multi-step sampling selects a future state $Y = s_{t+\tau}$ from the same trajectory of the sample $X = s_t$, where

$$\tau \sim q_{\gamma_{\mathrm{s}}}, \qquad q_{\gamma_{\mathrm{s}}}(h) = (1-\gamma_{\mathrm{s}})\gamma_{\mathrm{s}}^{h-1}, \qquad h = 1, 2, \ldots, \quad \gamma_{\mathrm{s}} \in [0, 1). \tag{7}$$

Here γ_{s} is the sampling discount factor. The multi-step sampling changes the transition kernel and the multi-step transition kernel is given by

$$P^\pi_{\gamma_{\mathrm{s}}}(u, dv) = \sum_{h=1}^\infty q_{\gamma_{\mathrm{s}}}(h)(P^\pi)^h(u, dv), \tag{8}$$

where $(P^\pi)^h$ denotes the h-step transition kernel. Setting $\gamma_{\mathrm{s}} = 0$ recovers one-step sampling.

3 Method

Here we introduce our approach, asymmetric Laplacian representations (ALR). Our goal is to learn good approximations of eigenvectors corresponding to low-magnitude eigenvalues of the asymmetric Laplacian $\mathcal{L} = I - \mathcal{P}$ via neural network learning in continuous and high-dimensional settings. Directly learning an ordered collection of complex eigenfunctions would require learning complex

values and phases during training. Instead, we learn a set of vectors that are real functions and require only that their span be approximately invariant under the same forward operator $\mathcal{P}$.

Since $\mathcal{L}$ is a combination of the identity operator I and $\mathcal{P}$, $\mathcal{L}$ and $\mathcal{P}$ share the same eigenvectors. For a subspace mixture of eigenvectors of $\mathcal{P}$, we have the closure:

$$\mathcal{P}F_\theta = F_\theta K, \qquad \mathcal{P}f_\theta = K^\top f_\theta, \qquad K \in \mathbb{R}^{d\times d}, \tag{9}$$

where F_θ is a mixture of eigenvectors, f_θ is a row vector $\in$ F, K represents the action of $\mathcal{P}$ in that basis. Equivalently, for every state, the conditional expectation of the successor is

$$(\mathcal{P}F_\theta)(x) = \mathbb{E}[F_\theta(Y) \mid X = x] = F_\theta(x)K. \tag{10}$$

If we have a set of vectors spanning an invariant subspace under $\mathcal{P}$, then they are mixtures of eigenvectors (which we can then unmix), provided the invariant subspace itself can be spanned by eigenvectors of $\mathcal{P}$. More specifically, $F = \text{span}_{\mathbb{C}}\{f_1, \ldots, f_m\} \subset \mathcal{H}$ is invariant if $\mathcal{P}F \subseteq F$, then for a vector f_i, either f_i is equal to a single eigenvector in $\mathcal{P}$ or f_i is a mixture, for example, $f_i = 0.1 * \phi_1^r + 0.4 * \phi_2^r + 0.5 * \phi_3^r$.

Our approach is motivated by the subspace iteration literature (Takeishi et al., 2017; Williams et al., 2015; Wu & Noé, 2020; Saad et al., 2000). Let $U_t \in \mathbb{R}^{N\times m}$ be a block of m vectors over $N = |S|$ finite states, where t denotes the iteration index. Starting from U_0 with sufficient rank and spectral overlap with the desired spectral region, subspace iteration repeatedly applies operations such as $U_{t+1} = PU_t$, with the aim of driving the span of U_t toward an invariant subspace of P containing the desired spectral directions. Exact multiplication by P changes the relative weights of existing spectral components but does not introduce components absent from U_0.

However, for a neural network based approach to subspace iteration, randomly initialized neural network outputs are not guaranteed to have sufficient spectral overlap with the desired eigenvectors of P. Furthermore, if, as we do partly in our approach, one approximates subspace iteration by training network outputs through an invariance loss, as we show in equation 12, rather than directly multiplying by P, gradient updates, generalization and function approximation can cause learnt vectors to lose linear independence, reducing the dimension of their span.

We therefore learn with both the invariance loss from equation 12 and an orthogonality loss that discourages the span from collapsing and encourages a sufficient span to overlap with the desired spectral directions of $\mathcal{P}$ associated with low magnitude eigenvalues. The orthogonality loss we obtain exactly from GDO equation 6 from Wu et al. (2019).

We split the problem into two phases. First, we learn vectors $F \in \mathbb{R}$ trained to be approximately invariant in $\mathcal{P}$ and have a rich span. Secondly, after learning a set of approximately invariant subspace vectors F, we assume that vectors F are approximate mixtures of eigenvectors. We then need to unmix the learnt subspace vectors, which we do without requiring access to $\mathcal{P}$ or reconstructing a graph. Instead, we unmix them by a small eigensolve of the reduced matrix K_{γ_K}. As an overview:

$$\underbrace{F}_{\approx \text{ mixed subspace vectors of } \mathcal{P}} \qquad \underbrace{\text{eigenvector } v_i \text{ of } (I - K_{\gamma_K})}_{\text{unmixing coefficients}} \implies \qquad \underbrace{Fv_i}_{\text{approximate Laplacian eigenfunctions}} \tag{11}$$

Phase 1: Learning real invariant subspace vectors F of $\mathcal{L}$. In the first phase, we learn a set of vectors $F \in \mathbb{R}$ whose span is approximately invariant under the forward operator $\mathcal{P}$, and hence under $\mathcal{L}$. If we had access to $\mathcal{P}$, we could formulate this as a mean-squared error based on the residual $\mathcal{P}F - FK$, and solving for F and K would give us vectors spanning an invariant subspace of $\mathcal{P}$. However, we do not have access to $\mathcal{P}$. In fact, we assume that we can access $\mathcal{P}$ only indirectly, through sampled transitions, potentially in high-dimensional continuous state spaces where no state is observed twice.

Notice, however, from equation 2, that Pf is a conditional expectation. We could therefore predict the features f of the next state given the current state using a predictor h of $\mathbb{E}[f(Y) \mid X]$. With an exact predictor, this would replace Pf, allowing us to learn the closure in equation 9 using the loss $\| \text{sg}(h) - K^\top f\|^2$, where $\text{sg}(\cdot)$ denotes a stop-gradient. However, any discrepancy between a learnt predictor h and the conditional expectation means that K is learning from an imperfect target. Since f changes during training, the predictor h may require multiple inner updates to estimate the changing conditional expectation accurately.

Therefore, instead of learning conditional invariance through a separate predictor h, we use a stopped target $f(Y)$, which gives the same gradient in expectation. The resulting loss is the mean-squared error between the stopped target $f(Y)$ and $K^\top f(X)$:

$$\mathcal{J}_{\text{inv}}(\theta, K) = \frac{1}{2n}\sum_{i=1}^{n}\left\|F_\theta(X_i)K - \text{sg}[F_\theta(Y_i)]\right\|_2^2. \tag{12}$$

In this phase, we optimize the invariance loss using samples drawn from episode trajectories. Pairs of a state X and a future state Y are obtained using the multi-step sampling from Wu et al. (2019), where a discount factor controls how far into the future we sample Y. Since our method needs only an invariance term and orthogonality to train the subspace vectors F, we combine the invariance term with the orthogonality and norm loss from Section 2,

$$\mathcal{J}_{\text{subspace}}(\theta, K) = \mathcal{J}_{\text{inv}}(\theta, K) + \beta\mathcal{J}_{\text{orth}}(\theta) \tag{13}$$

Phase 2: Unmixing subspace vectors F via multi-step K_{γ_K}. Given a set of vectors F_θ learnt using equation equation 9, whose span is approximately invariant and sufficiently rich to approximate our desired directed eigenfunctions (as encouraged by the orthogonality and norm loss J_{orth}), we unmix the vectors to obtain approximations of our desired asymmetric eigenfunctions, taking an approach from the projection method literature (Saad et al., 2000). For our learnt vectors F_θ, first consider the ideal case in which an eigenfunction ψ of the asymmetric Laplacian lies in this subspace. Then there is a coefficient vector $v \in \mathbb{C}^d$ such that $\psi = F_\theta v$. We can then express $\mathcal{L}\psi = \lambda\psi$ as $\mathcal{L}F_\theta v = \lambda F_\theta v$. Multiplying both sides by F_θ^* gives $F_\theta^*\mathcal{L}F_\theta v = \lambda F_\theta^* F_\theta v = \lambda M_F v$. Here $M_F = F_\theta^* F_\theta$ is the Gram matrix defined in Section 2, and F_θ^* denotes the adjoint. For the one-step derivation below, let K_1 denote the reduced matrix associated with $\mathcal{P}$; the matrix K fitted in Phase 1 instead reflects the sampling regime used during training. Starting from the closure in equation 9, under exact invariance, we have

$$\mathcal{L}F_\theta = (I - \mathcal{P})F_\theta = F_\theta(I_d - K_1). \tag{13}$$

Set $A = F_\theta^*\mathcal{L}F_\theta$. Assuming the columns of F_θ are linearly independent in $L^2(\rho)$, we obtain

$$\begin{aligned} A &= F_\theta^*(I - \mathcal{P})F_\theta = F_\theta^*F_\theta - F_\theta^*(\mathcal{P}F_\theta) = M_F - M_F K_1 = M_F(I_d - K_1), \\ Av = \lambda M_F v &\iff M_F(I_d - K_1)v = \lambda M_F v \\ &\iff (I_d - K_1)v = \lambda v. \end{aligned} \tag{14}$$

Thus, given equation 14, eigensolving $(I_d - K)$ gives us our estimated solutions for the coefficients v needed to estimate the asymmetric eigenvectors via $\psi = F_\theta v$ and their eigenvalues λ.

Here we assume that we have learned a subspace of P approximately well with a set of subspace vectors F that are approximate mixtures of asymmetric eigenvectors of P. After the subspace F learning phase, with the matrices F and K, as discussed in Section 3, we can perform a small eigensolve over the reduced K matrix (without any samples over the space), and obtain approximations of eigenvectors. However, in the prior invariance loss equation 12, whilst $F(Y)$ has a stop-gradient, $F(X)$ is moving, thus K is fitted between two likely different F vectors over the space. The F vectors that we use to estimate the K matrices are the final vectors after training. To estimate K for this fixed representation, we freeze $F(X)$ and $F(Y)$ and learn only K within the invariance loss.

However, domains can be very large or very small, which can strongly affect the eigenvalue scales and thus the reduced matrices. For small domains, it may be sufficient to consider the one-step version of our invariance loss in equation 12, where Y is the state immediately after X. There is a trade-off between one-step and multi-step learning. Vijayan et al. (2026) discussed how eigenfunctions of $\mathcal{P}$ remain eigenfunctions of its powers, with transformed eigenvalues. This motivates learning the subspace vectors F_θ with a multi-step invariance loss and then, by freezing F_θ, learning a set of matrices K_h under different sampling procedures, estimating eigenvalues for different horizons. A weighted combination of these matrices provides a reduced approximation of the corresponding weighted multi-step operator, including the one-step case when that horizon is included. Thus, we can learn with a multi-step update during the subspace-learning phase and estimate one-step solutions afterwards, which is empirically most important for smaller domains or where intricate directed or stochastic differences between states occur.

As discussed by Vijayan et al. (2026), a one-step Laplacian eigenfunction f with eigenvalue λ satisfies $\mathcal{P}^h f = (1 - \lambda)^h f$. Under exact closure in equation (9), we also have $\mathcal{P}^h F_\theta = F_\theta K_1^h$. Here K_1^h

denotes a matrix power, whereas K_h below denotes a matrix fitted independently for horizon h. If Y_h is sampled h steps after X, then $(\mathcal{P}^h f)(X) = \mathbb{E}[f(Y_h) \mid X]$. During our K-learning phase, we learn multiple matrices K_h that fit $F_\theta(X)K_h$ to the future features $F_\theta(Y_h)$ for a set of fixed horizons $h \in H$, with the loss

$$\mathcal{J}_{\text{horizon}}(K_h) = \frac{|H|}{2n} \sum_{i=1}^{n/|H|} \left\| \text{sg}[F_\theta(X_{h,i})]K_h - \text{sg}[F_\theta(Y_{h,i})] \right\|_2^2 . \tag{15}$$

Here $Y_{h,i}$ is sampled h steps after $X_{h,i}$ within a trajectory, and n is the total number of samples, split equally across horizons. No orthogonality term is optimised in this phase because F_θ is frozen. The full Phase 2 objective sums equation 15 over $h \in H$: $\mathcal{J}_{\text{Phase 2}}\big(\{K_h\}_{h\in H}\big) = \sum_{h\in H} \mathcal{J}_{\text{horizon}}(K_h)$.

Lastly, we combine the matrices K_h across horizons using a discount factor $\gamma_K \in (0, 1]$:

$$K_{\gamma_K} = \sum_{h\in H} w_h(\gamma_K) K_h, \qquad w_h(\gamma_K) = \frac{\gamma_K^{h-1}}{\sum_{j\in H} \gamma_K^{j-1}}. \tag{16}$$

This is a quantized approximation of the reduced matrix associated with the geometrically weighted multi-step sampling used in Phase 1. To include $\gamma_K = 0$, provided $1 \in H$, define $w_1(0) = 1$ and $w_h(0) = 0$ for $h > 1$. This recovers the one-step sampling case, making only K_1 contribute to K_{γ_K}. The choice $\gamma_K = 1$ gives the arithmetic mean, whilst values between zero and one give more weight to matrices K_h obtained under shorter horizons. With K_{γ_K}, we then unmix our subspace vectors F_θ by obtaining coefficients from the small eigensolve

$$(I_d - K_{\gamma_K})R = R\Lambda_{\gamma_K}. \tag{17}$$

We approximate the right eigenfunctions with $\widehat{\Phi^r} = F_\theta R$, the left eigenfunctions with $\widehat{\Phi^l} = F_\theta(R^{-1})^*$ and associated eigenvalues with Λ_{γ_K}, where the i_{th} diagonal value is the approximated eigenvalue of the i_{th} row $\approx$ eigenvector in F_θ. We use $\widehat{\phi^r}$ to denote an vector in $\widehat{\Phi^r}$ and $\widehat{\phi^l}$ to denote an vector in $\widehat{\Phi^l}$.

4 Evaluation

Grid domains. We evaluate ALR in four grid domains of varying sizes and structures, using state observations $x \in \mathbb{R}$ and $y \in \mathbb{R}$. These domains are a directed clockwise `cycle` that consists of 200 states, the directed `Four Rooms` domain (Theophilou & Şimşek, 2026) scaled to five times its original size, the symmetric `Grid64` domain (Gomez et al., 2024), and a `directed Grid64` domain constructed by adding five directed teleportation edges between states that are furthest away from each other, in the following manner: We find the two states that are furthest away in shortest path distance; if there is a tie, we pick the pair with the smallest node index; we create a directed edge from the lowest node value in that pair to the highest; we perform this process five times, creating directed edges across distant states. In all four domains, we collect samples following the procedure by Gomez et al. (2024). Specifically, an agent starts in a random state, does a random walk for n steps, then repeats the same process starting in another random state. This process is used to encourage coverage of the state space under a limited trajectory horizon. Like Gomez et al. (2024), we capture one million transition samples per grid domain, which we keep fixed across domains for consistency.

Drone domains and image observations. We also use four drone domains constructed using Gym-PyBullet-Drones (Panerati et al., 2021). A visualization is included in Appendix D. Each domain has stochastic wind throughout the space, pushing the agent along flow directions. The geometric structure of these domains is three-dimensional. We design the wind in two dimensions and then extend it to the three-dimensional setting by repeating the same wind flow pattern along the third (vertical) axis. We evaluate three versions of each drone domain, with different state observations. The first version uses raw physical observations provided by the simulator. For the other two observation types, we use images: a top-down view where the drone has a shadow, which provides a visual cue for its height, and an isometric view. In each drone domain, we use 327,680 samples.

Parameters. Learning for each method is performed with 16 random seeds per domain and observation type. We do 100,000 iterations of Phase 1, where we learn the subspace vectors F_θ with learning rate 0.001 for numerical observations and 0.0003 for image observations. During phase 1 in ALR, trajectory samples use the standard multi-step sampling from GDO with discount factor 0.9 (Wu

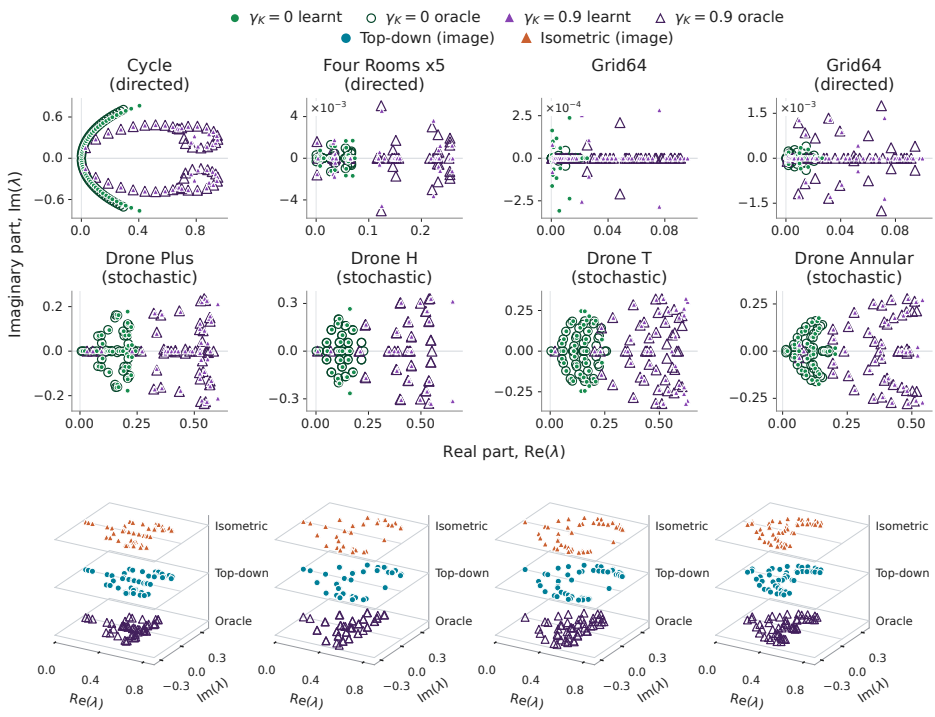

Figure 1: Learned (with ALR) and oracle eigenvalues in all domains, for $\gamma_K = 0$ and $\gamma_K = 0.9$.

et al., 2019). For phase 2 in ALR, we do 50,000 iterations, where a set of K_h matrices is learnt with learning rate 0.0001 while keeping F_θ frozen. ALLO uses 150,000 joint training iterations, a learning rate of 0.001 and a sampling discount of 0.9. Full details are given in Table 1 in the appendix.

Oracle eigenvector and eigenvalues. We have specifically considered domains for which we are able to obtain oracle eigenvectors and eigenvalues. Since $\mathcal{P}$ is an operator induced by policy π, the oracle must reflect the sample transitions obtained by the policy. We start with transition matrix P^π for a domain; initially, a random walk transition matrix. Then we count every transition in the training trajectories and use this as weight for the transition probabilities in P^π. Using P^π, we construct $L = I - P^\pi$ and take its lowest magnitude eigenvalues and associated eigenvectors. In the drone domains, we use 880 voxels as discretized states, which we obtain P^π from (adjacent voxels are connected as edges), and similarly, count the sample transitions to weight the edges.

Eigenvalue recovery. We first evaluate ALR's recovery of the low-magnitude eigenvalues and corresponding left and right eigenvectors. Figure 1 compares learned and oracle eigenvalues across domains for $\gamma_K = 0$ and $\gamma_K = 0.9$. ALR closely recovers the complex spectra under both settings and captures the changes induced by the multi-step discount. Although Grid64 is symmetric in structure, its sampling policy is not, resulting in small non-zero oracle imaginary components. In the drone domains, both top-down and isometric image observations recover the overall shape of the low-magnitude complex spectrum.

Cosine similarity. Figure 2 shows cosine similarity for $\gamma_K \in \{0, 0.5, 0.9, 0.99, 0.995\}$. Larger discounts perform poorly in the 200-state directed cycle, whereas in Grid64 and the drone domains they are generally competitive with, or better than, smaller discounts. Conversely, $\gamma_K = 0$ performs poorly in Grid64, where longer-range propagation is important. The drone domains are less affected because stochastic wind induces additional state-space propagation along trajectories. Recovery typically stabilizes within approximately 20,000 K-learning updates. Since F_θ is frozen during this phase and only the small K_h matrices are optimized, these updates are substantially cheaper than Phase 1 representation learning. The table on the next page shows detailed results on ALR and comparisons with ALLO, except for the image-based observations, which are shown in Appendix A. We see that ALR substantially improves over ALLO in the directed and stochastic domains, particularly

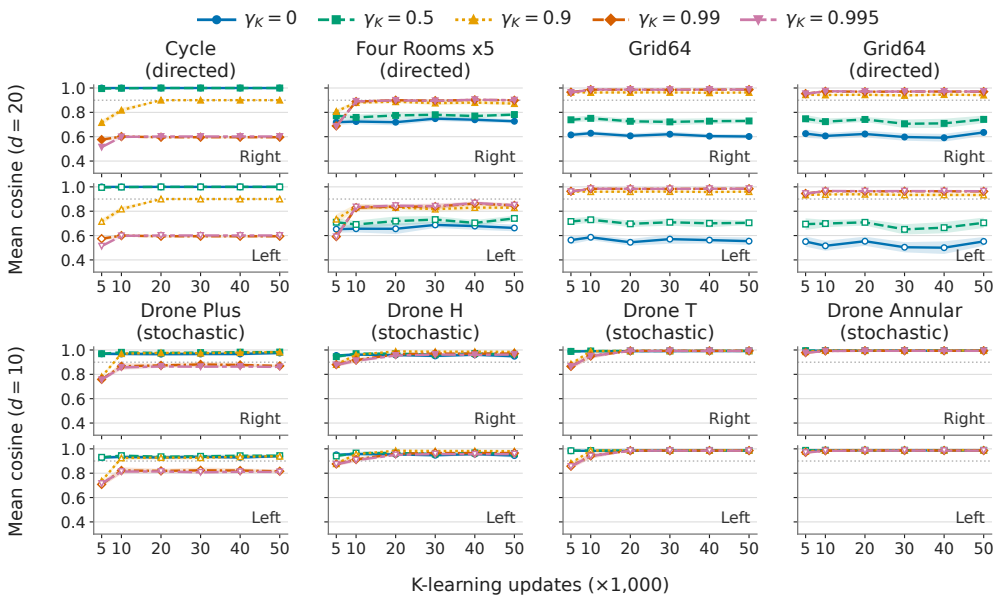

Figure 2: Cosine similarity (with ALR) in all domains, for $\gamma_K \in \{0, 0.5, 0.9, 0.99, 0.995\}$.

in the drone environments where ALLO's cosine similarity degrades rapidly as d increases. ALR generally achieves cosine similarity above 0.9 for $d = 5$, and frequently for $d = 10$ and $d = 20$, with comparable recovery of left and right eigenvectors. For image observations, both top-down and isometric views exceed 0.98 cosine similarity at $d = 5$ across all drone domains, while performance remains strongest in Drone Plus and Drone Annular as d increases. Together with the low eigenvalue error in Figure 1, these results show that ALR accurately recovers both sides of the low-magnitude asymmetric eigensystem from raw and high-dimensional image observations.

Hitting time distance and stationary distribution estimation. The left eigenvector associated with the lowest magnitude eigenvalue is equivalently the stationary distribution since $\lambda = 0$. This allows us to obtain two additional properties. First, we get an approximation of the stationary distribution we denote as $\widehat{\rho}$. Secondly, we can use the existing literature on hitting time estimation from asymmetric eigenvectors and eigenvalues (Meyer, 1975) and estimate truncated hitting time distances with: $\widehat{h}_d(i,j) = \frac{1}{\widehat{\phi}^l_k(j)} \sum_{k=1}^{d} \frac{(\widehat{\phi}^r_k(j) - \widehat{\phi}^r_k(i))\widehat{\phi}^l_k(j)}{\widehat{\lambda}_k}$. In Figure 3, we show, in all domains, with a Spearman metric, comparisons between this hitting time distance in state to goal pairs against ALLO's commute-times (RA-LapRep, Wang et al. (2023b)) estimated with the same eigenvector dimensionality of 50. We see clearly better performance for asymmetric distance learning from ALR compared to ALLO across domains. The figure also shows the cosine similarity between oracle stationary distributions we denote with ρ and the estimated left eigenvector $\widehat{\rho}$ with a cosine similarity score above 0.9.

5 Conclusion and Future Work

We proposed an approach to estimating eigenvectors and eigenvalues of asymmetric Laplacian, evaluating against the previous state-of-the-art baseline (ALLO), which assumes symmetric Laplacian operators. We found that, by learning a joint invariant and orthogonality loss, we could obtain vectors that can be rotated to approximate well the oracle eigenvectors in directed and stochastic domains, with both vector state inputs and high-dimensional images. Additionally, we found that asymmetric distances can be obtained from ALR, which improve in directed and stochastic settings against the baseline RALapRep. An potential extension of ALR, given the stationary distribution results, is to explore the stationary distribution estimates for exploration learning with related literature (Zhang et al., 2021; Jiang et al., 2025; Yang et al., 2024; Strehl & Littman, 2008; Lobel et al., 2023; Şimşek & Barto, 2004), given that these exploration methods utilize counts and novelty for intrinsic motivation.

Method	Metric	$d = 5$	$d = 10$	$d = 20$	$d = 50$	$d = 5$	$d = 10$	$d = 20$	$d = 50$
		Cycle (directed)				**Drone Plus**			
ALR	Left cosine ↑	.799965(12)	.999906(12)	.900010(84)	.964(18)	.9674(16)	.939(18)	.898(28)	.743(15)
ALR	Right cosine ↑	.799967(12)	.999907(12)	.90004(10)	.964(17)	.99575(96)	.9825(96)	.911(30)	.8543(98)
ALR	Eigenvalue MAE ↓	.145712(24)	.000192(32)	.026(11)	.0067(31)	.00278(29)	.00412(26)	.0095(58)	.0178(18)
ALLO	Right cosine ↑	.706845(54)	.70652(13)	.70530(25)	.6863(77)	.749(14)	.536(14)	.359(11)	.227(13)
ALLO	Eigenvalue MAE ↓	.147548(78)	.29946(18)	.51491(30)	.76147(24)	.0718(37)	.1261(48)	.1855(89)	.265(26)
		Grid64				**Drone T**			
ALR	Left cosine ↑	.939(33)	.959(18)	.961(13)	.925(10)	.99199(57)	.9884(13)	.793(13)	.7559(83)
ALR	Right cosine ↑	.944(32)	.962(17)	.963(13)	.929(10)	.99627(23)	.99374(85)	.799(14)	.8102(73)
ALR	Eigenvalue MAE ↓	.000192(35)	.000253(36)	.000390(39)	.00104(11)	.00153(29)	.00188(24)	.0324(11)	.0352(16)
ALLO	Right cosine ↑	.864(42)	.909(20)	.872(10)	.8710(95)	.606(26)	.409(16)	.298(31)	.187(18)
ALLO	Eigenvalue MAE ↓	.001515(36)	.003393(52)	.010441(85)	.04614(22)	.1107(17)	.1567(26)	.2050(39)	.2583(68)
		Grid64 (directed)				**Drone Annular**			
ALR	Left cosine ↑	.905(68)	.912(38)	.933(23)	.890(20)	.99199(29)	.98926(35)	.93316(94)	.866(13)
ALR	Right cosine ↑	.898(51)	.923(27)	.942(20)	.906(19)	.99688(16)	.99592(22)	.94131(87)	.889(13)
ALR	Eigenvalue MAE ↓	.000252(39)	.000325(33)	.000400(47)	.000983(83)	.00350(26)	.00318(19)	.00822(17)	.0257(76)
ALLO	Right cosine ↑	.555(18)	.6504(89)	.6391(70)	.6461(90)	.466(18)	.4793(49)	.3923(42)	.2672(47)
ALLO	Eigenvalue MAE ↓	.002984(94)	.00512(12)	.01272(17)	.04957(38)	.05940(27)	.09545(42)	.14079(69)	.2165(14)
		Four Rooms ×5 (directed)				**Drone H**			
ALR	Left cosine ↑	.930(38)	.780(48)	.829(21)	.756(26)	.9851(49)	.9812(79)	.8593(53)	.8263(68)
ALR	Right cosine ↑	.932(54)	.855(40)	.872(21)	.808(18)	.9918(39)	.9848(80)	.8582(49)	.8596(80)
ALR	Eigenvalue MAE ↓	.000345(70)	.00091(17)	.000822(89)	.00146(14)	.00195(19)	.00149(15)	.03278(21)	.0347(35)
ALLO	Right cosine ↑	.6783(43)	.6185(34)	.744(12)	.6446(91)	.613(15)	.501(10)	.3160(48)	.2058(59)
ALLO	Eigenvalue MAE ↓	.016426(30)	.028221(34)	.064565(40)	.15744(13)	.08236(39)	.12591(45)	.17315(88)	.2384(15)

Cosine: mean absolute eigenvector cosine similarity (↑). MAE: mean absolute eigenvalue error (↓). In the parenthesis, we show the standard deviation in the last displayed digits.

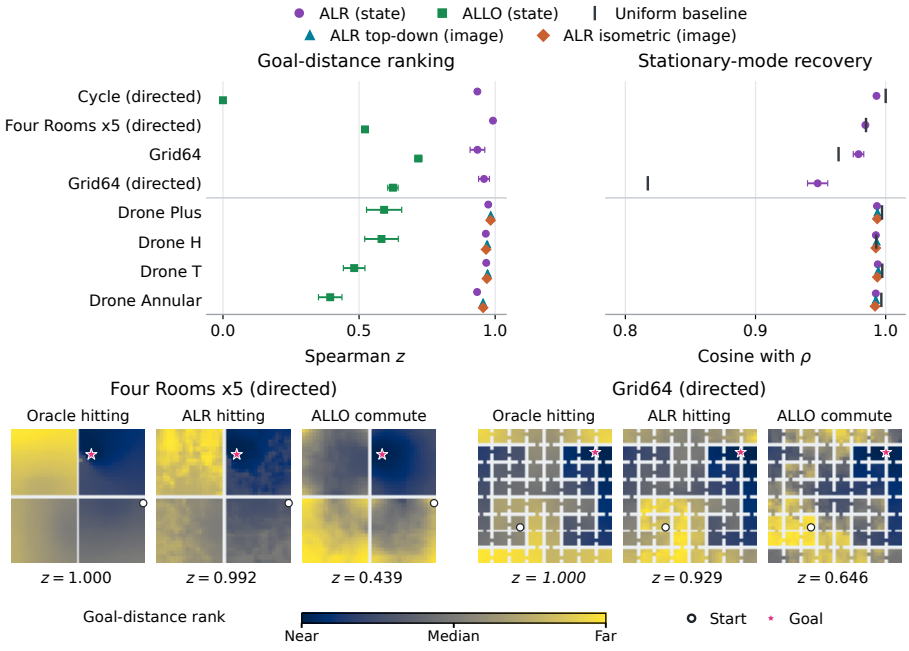

Figure 3: The goal-distance ranking plot (top left) shows the Spearman metric for ALR hitting-time distance representations and ALLO commute-time distances; higher is better. The stationary-mode recovery plots shows the cosine similarity between the true stationary distributions ρ and ALR estimated stationary distributions $\widehat{\rho}$; higher is better. In the bottom plots, we visual the distances in the goal-distance cases where z shows the Spearman score.

REFERENCES

Diego Gomez, Michael Bowling, and Marlos C. Machado. Proper laplacian representation learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=7gLfQT52Nn.

Yuhua Jiang, Qihan Liu, Yiqin Yang, Xiaoteng Ma, Dianyu Zhong, Hao Hu, Jun Yang, Bin Liang, Bo XU, Chongjie Zhang, and Qianchuan Zhao. Episodic novelty through temporal distance. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=I7DeajDEx7.

Martin Klissarov and Marlos C Machado. Deep Laplacian-based options for temporally-extended exploration. In *International Conference on Machine Learning*, pp. 17198–17217. PMLR, 2023.

Hebin Liang, Yi Ma, Chenjun Xiao, Zibin Dong, Zilin Cao, Fei Ni, Yifu Yuan, and Jianye HAO. Bottleneck-guided spectral subgoals for offline goal-conditioned RL. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=l4Q8dGGHVC.

Sam Lobel, Akhil Bagaria, and George Konidaris. Flipping coins to estimate pseudocounts for exploration in reinforcement learning. In *International Conference on Machine Learning*, pp. 22594–22613. PMLR, 2023.

Carl D. Meyer, Jr. The role of the group generalized inverse in the theory of finite markov chains. *SIAM Review*, 17(3):443–464, 1975. doi: 10.1137/1017044. URL https://doi.org/10.1137/1017044.

Jacopo Panerati, Hehui Zheng, Siqi Zhou, James Xu, Amanda Prorok, and Angela P. Schoellig. Learning to fly - a gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control. *CoRR*, abs/2103.02142, 2021. URL https://arxiv.org/abs/2103.02142.

Yousef Saad, Richard B. Lehoucq, and Danny C. Sorensen. An introduction to iterative projection methods. In Zhaojun Bai, James Demmel, Jack J. Dongarra, Axel Ruhe, and Henk A. van der Vorst (eds.), *Templates for the Solution of Algebraic Eigenvalue Problems*, volume 11 of *Software, environments, tools*, pp. 37–44. SIAM, 2000. doi: 10.1137/1.9780898719581.CH3. URL https://doi.org/10.1137/1.9780898719581.ch3.

Dikshant Shehmar, Matthew Kyle Schlegel, Matthew E. Taylor, and Marlos C. Machado. Laplacian representations for decision-time planning. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=6JwkAlGLjT.

Özgür Şimşek and Andrew G Barto. Using relative novelty to identify useful temporal abstractions in reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine Learning*, pp. 95, 2004.

Alexander L Strehl and Michael L Littman. An analysis of model-based interval estimation for Markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. Learning koopman invariant subspaces for dynamic mode decomposition. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3a835d3215755c435ef4fe9965a3f2a0-Paper.pdf.

Andreas Theophilou and Özgür Şimşek. Novel exploration via orthogonality. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=yJS1eZSNUv.

Priyesh Vijayan, Élodie Côté-Gauthier, Mathieu Reymond, Sarath Chandar, Doina Precup, and Isabeau Prémont-Schwarz. Addressing sparse-rewards in RL with scalable hierarchical novel eigen options. In *Third Reinforcement Learning Beyond Rewards Workshop at the Reinforcement Learning Conference*, 2026. URL https://openreview.net/forum?id=Bn5TzXbC6s.

Kaixin Wang, Kuangqi Zhou, Qixin Zhang, Jie Shao, Bryan Hooi, and Jiashi Feng. Towards better Laplacian representation in reinforcement learning with generalized graph drawing. In *International Conference on Machine Learning*, pp. 11003–11012. PMLR, 2021.

Kaixin Wang, Kuangqi Zhou, Jiashi Feng, Bryan Hooi, and Xinchao Wang. Reachability-aware laplacian representation in reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 36670–36693, 2023a.

Kaixin Wang, Kuangqi Zhou, Jiashi Feng, Bryan Hooi, and Xinchao Wang. Reachability-aware laplacian representation in reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023b.

{Matthew O.} Williams, Yannis Kevrekidis, and {Clarence Worth} Rowley. A data–driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, December 2015. ISSN 0938-8974. doi: 10.1007/s00332-015-9258-5. Publisher Copyright: © 2015, Springer Science+Business Media New York.

Hao Wu and Frank Noé. Variational approach for learning markov processes from time series data. *J. Nonlinear Sci.*, 30(1):23–66, 2020. doi: 10.1007/S00332-019-09567-Y. URL `https://doi.org/10.1007/s00332-019-09567-y`.

Yifan Wu, George Tucker, and Ofir Nachum. The Laplacian in RL: Learning representations with efficient approximations. In *International Conference on Learning Representations*, 2019.

Kai Yang, Jian Tao, Jiafei Lyu, and Xiu Li. Exploration and anti-exploration with distributional random network distillation. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 56397–56421, 2024.

Tianjun Zhang, Huazhe Xu, Xiaolong Wang, Yi Wu, Kurt Keutzer, Joseph E Gonzalez, and Yuandong Tian. Noveld: A simple yet effective exploration criterion. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 25217–25230. Curran Associates, Inc., 2021.

A Image-based eigenfunction cosine similarities

		Drone Plus				**Drone H**			
Observation	Metric	$d = 5$	$d = 10$	$d = 20$	$d = 50$	$d = 5$	$d = 10$	$d = 20$	$d = 50$
Image-Top-down	Left cosine ↑	.99248(77)	.9838(18)	.7770(72)	.6594(68)	.9864(41)	.7953(52)	.7639(38)	.6254(23)
	Right cosine ↑	.99699(99)	.9887(21)	.8067(28)	.6790(51)	.9945(21)	.8032(61)	.7697(35)	.6305(23)
	Eigenvalue MAE ↓	.00138(22)	.00243(31)	.02535(90)	.0646(14)	.00105(19)	.01671(30)	.05677(27)	.1212(30)
Image-Isometric	Left cosine ↑	.99181(88)	.9830(16)	.7756(61)	.6599(58)	.9863(37)	.7947(48)	.7649(43)	.6260(20)
	Right cosine ↑	.9969(11)	.9884(20)	.8053(33)	.6791(63)	.9944(23)	.8012(51)	.7710(40)	.6313(19)
	Eigenvalue MAE ↓	.00177(23)	.00265(35)	.02560(97)	.0653(17)	.00174(26)	.01754(37)	.05731(32)	.1184(58)
		Drone T				**Drone Annular**			
Image-Top-down	Left cosine ↑	.99310(44)	.80891(82)	.8003(17)	.52634(64)	.99226(52)	.99012(43)	.93678(76)	.8852(23)
	Right cosine ↑	.99694(18)	.82553(56)	.8380(18)	.54600(81)	.99769(29)	.99705(24)	.9434(12)	.9061(22)
	Eigenvalue MAE ↓	.00167(40)	.01779(63)	.04542(34)	.1218(50)	.00164(32)	.00148(31)	.00682(30)	.02206(26)
Image-Isometric	Left cosine ↑	.99280(40)	.813(21)	.803(12)	.5281(59)	.987(10)	.9874(58)	.9358(32)	.881(13)
	Right cosine ↑	.99719(15)	.831(20)	.841(11)	.5479(83)	.99811(18)	.99753(16)	.9445(10)	.901(15)
	Eigenvalue MAE ↓	.00109(42)	.0174(19)	.0439(68)	.1233(92)	.00237(37)	.00219(29)	.00747(29)	.0250(39)

B Visual cosine similarities between ALR and oracle eigenvectors

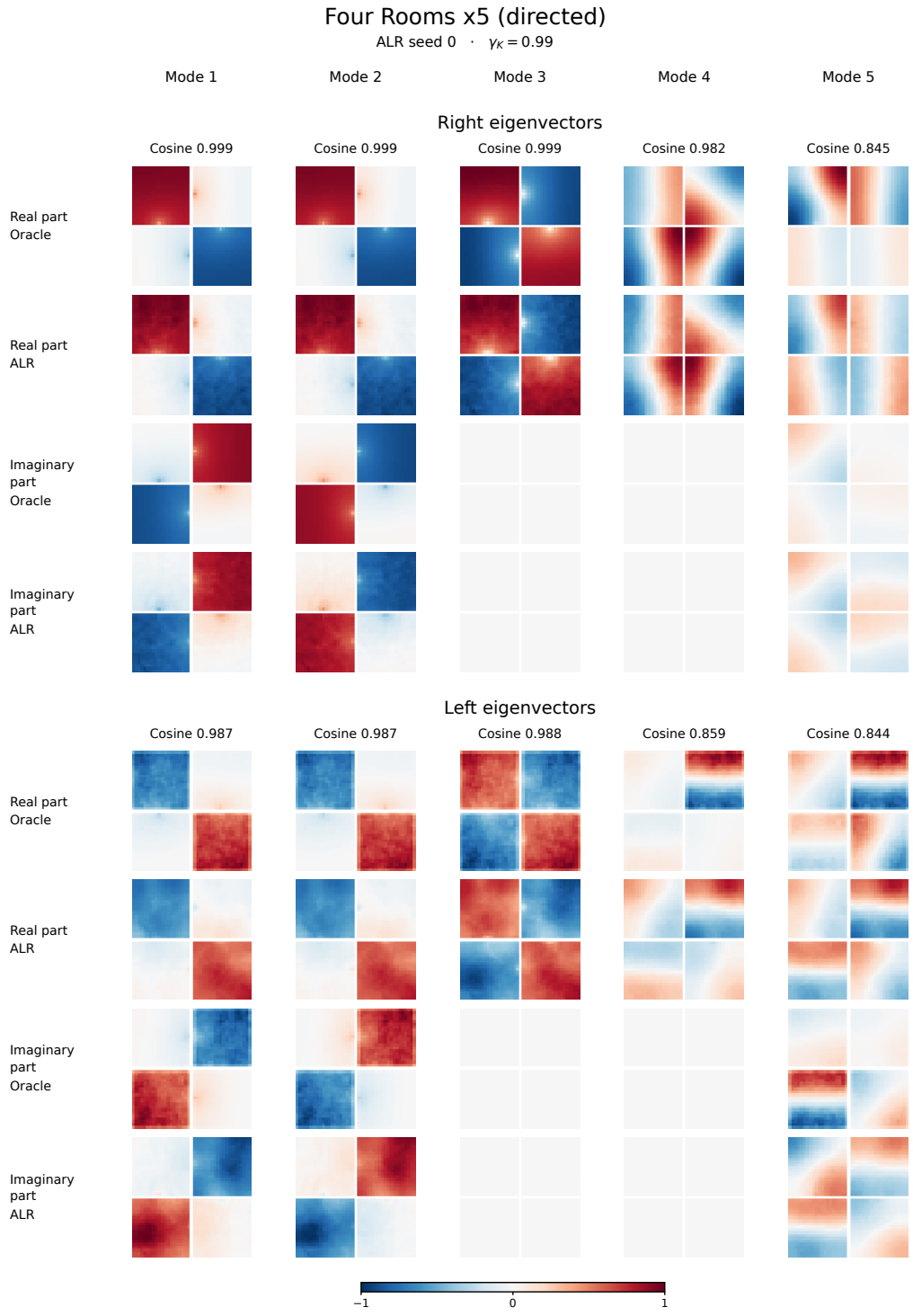

Figure 4

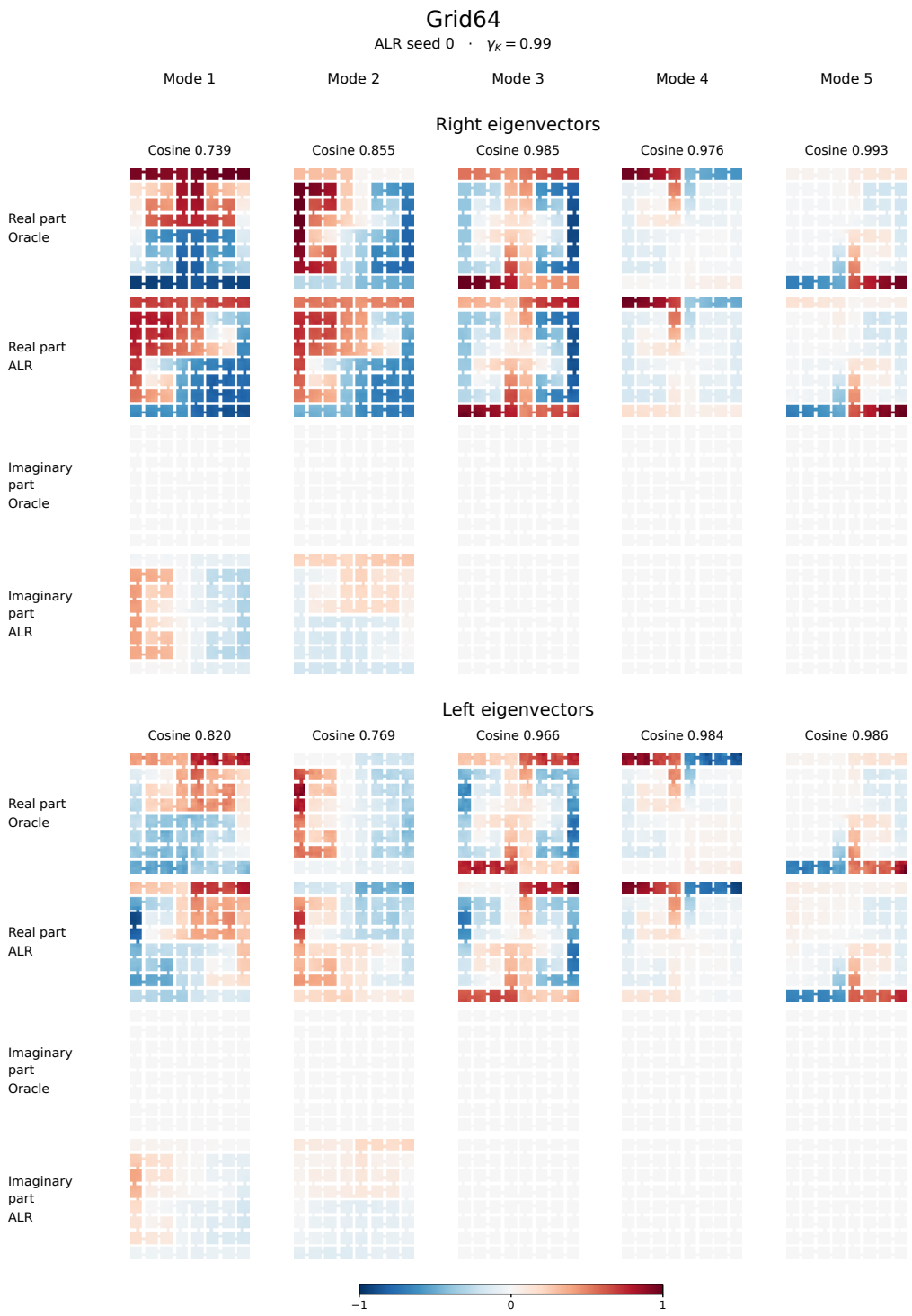
Grid64
ALR seed 0 · $\gamma_K = 0.99$
Mode 1
Mode 2
Mode 3
Mode 4
Mode 5
Right eigenvectors
Cosine 0.739
Cosine 0.855
Cosine 0.985
Cosine 0.976
Cosine 0.993
Real part Oracle
Real part ALR
Imaginary part Oracle
Imaginary part ALR
Left eigenvectors
Cosine 0.820
Cosine 0.769
Cosine 0.966
Cosine 0.984
Cosine 0.986
Real part Oracle
Real part ALR
Imaginary part Oracle
Imaginary part ALR
−1
0
1

Figure 5

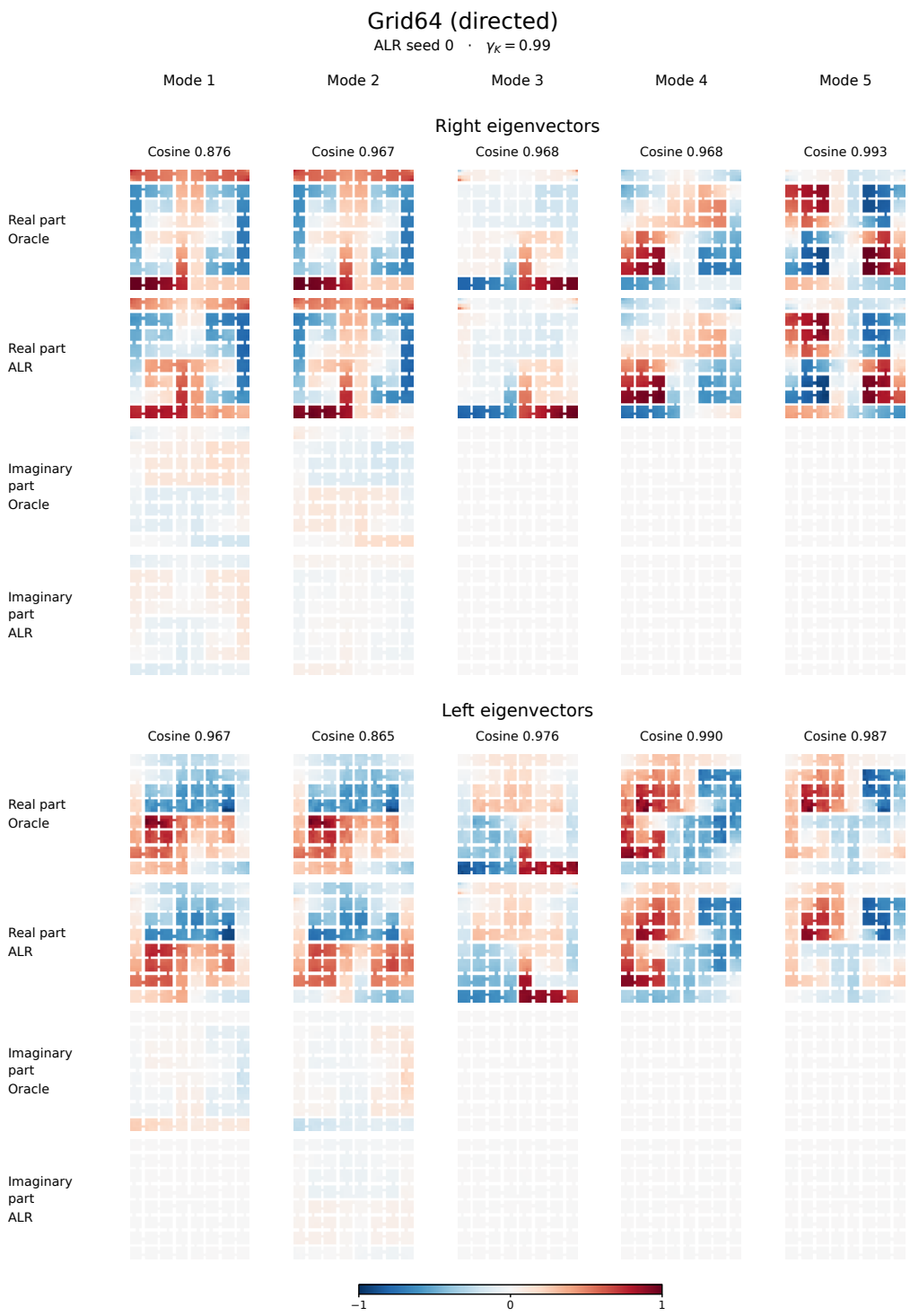
Grid64 (directed)
ALR seed 0 · $\gamma_K = 0.99$
Mode 1
Mode 2
Mode 3
Mode 4
Mode 5
Right eigenvectors
Cosine 0.876
Cosine 0.967
Cosine 0.968
Cosine 0.968
Cosine 0.993
Real part Oracle
Real part ALR
Imaginary part Oracle
Imaginary part ALR
Left eigenvectors
Cosine 0.967
Cosine 0.865
Cosine 0.976
Cosine 0.990
Cosine 0.987
Real part Oracle
Real part ALR
Imaginary part Oracle
Imaginary part ALR
−1
0
1

Figure 6

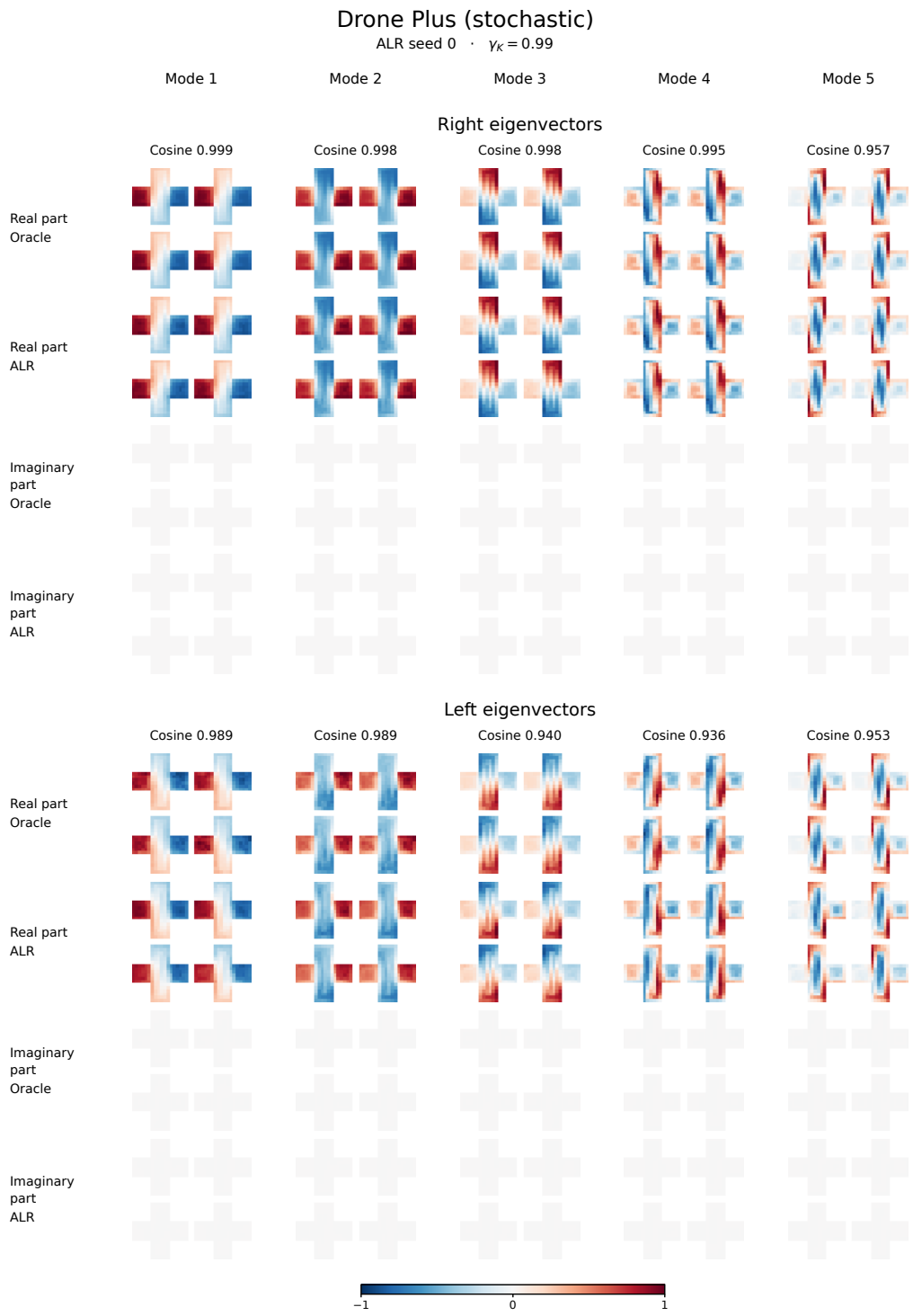
Drone Plus (stochastic)
ALR seed 0 · $\gamma_K = 0.99$
Mode 1
Mode 2
Mode 3
Mode 4
Mode 5
Right eigenvectors
Cosine 0.999
Cosine 0.998
Cosine 0.998
Cosine 0.995
Cosine 0.957
Real part Oracle
Real part ALR
Imaginary part Oracle
Imaginary part ALR
Left eigenvectors
Cosine 0.989
Cosine 0.989
Cosine 0.940
Cosine 0.936
Cosine 0.953
Real part Oracle
Real part ALR
Imaginary part Oracle
Imaginary part ALR
−1
0
1

Figure 7


Drone H (stochastic)
ALR seed 0 · $\gamma_K = 0.99$
Mode 1
Mode 2
Mode 3
Mode 4
Mode 5
Right eigenvectors
Cosine 0.999
Cosine 0.999
Cosine 0.985
Cosine 0.989
Cosine 0.972
Real part Oracle
Real part ALR
Imaginary part Oracle
Imaginary part ALR
Left eigenvectors
Cosine 0.994
Cosine 0.990
Cosine 0.982
Cosine 0.977
Cosine 0.961
Real part Oracle
Real part ALR
Imaginary part Oracle
Imaginary part ALR
−1
0
1

Figure 8

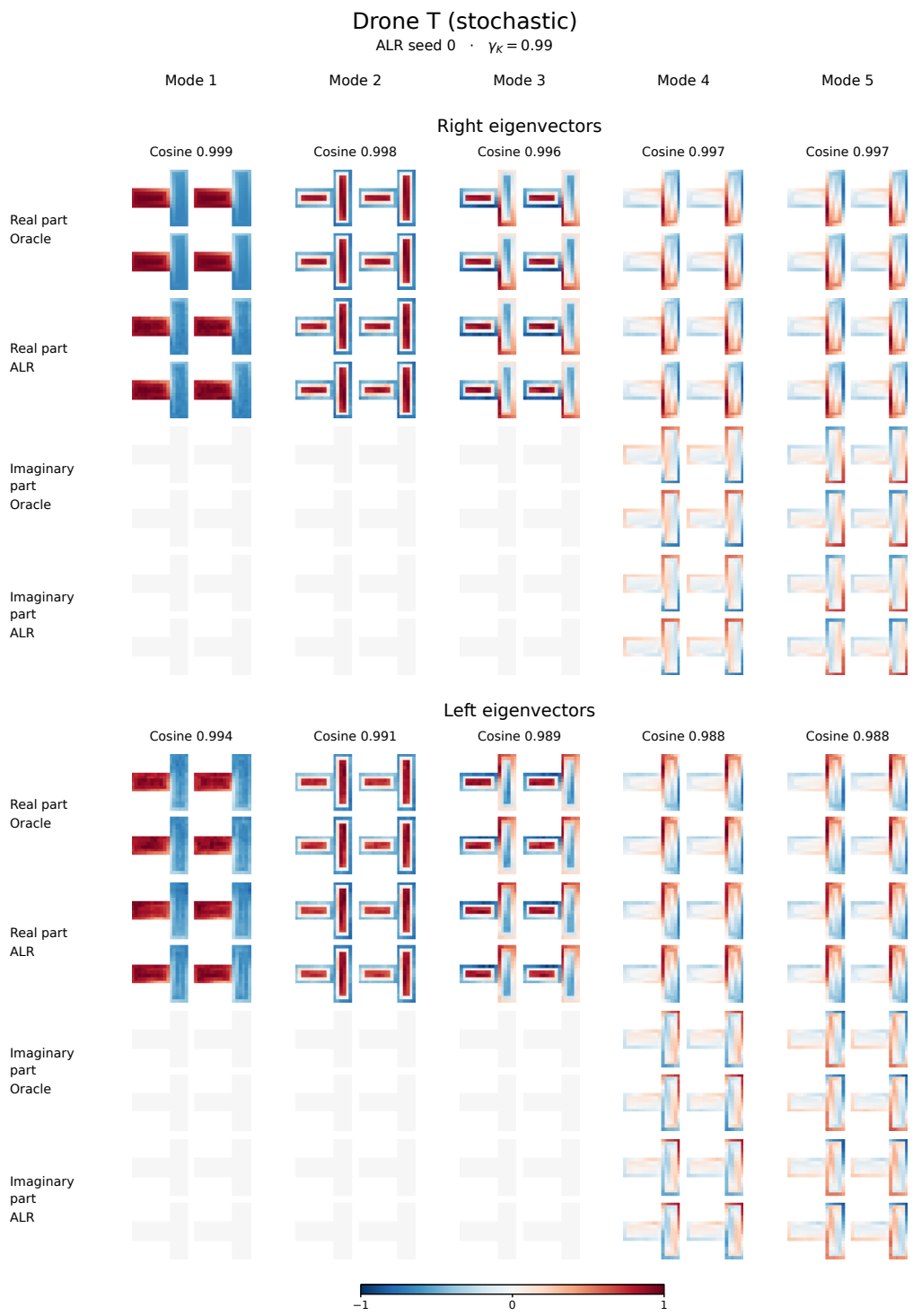
Drone T (stochastic)
ALR seed 0 · $\gamma_K = 0.99$
Mode 1
Mode 2
Mode 3
Mode 4
Mode 5
Right eigenvectors
Cosine 0.999
Cosine 0.998
Cosine 0.996
Cosine 0.997
Cosine 0.997
Real part Oracle
Real part ALR
Imaginary part Oracle
Imaginary part ALR
Left eigenvectors
Cosine 0.994
Cosine 0.991
Cosine 0.989
Cosine 0.988
Cosine 0.988
Real part Oracle
Real part ALR
Imaginary part Oracle
Imaginary part ALR
−1
0
1

Figure 9

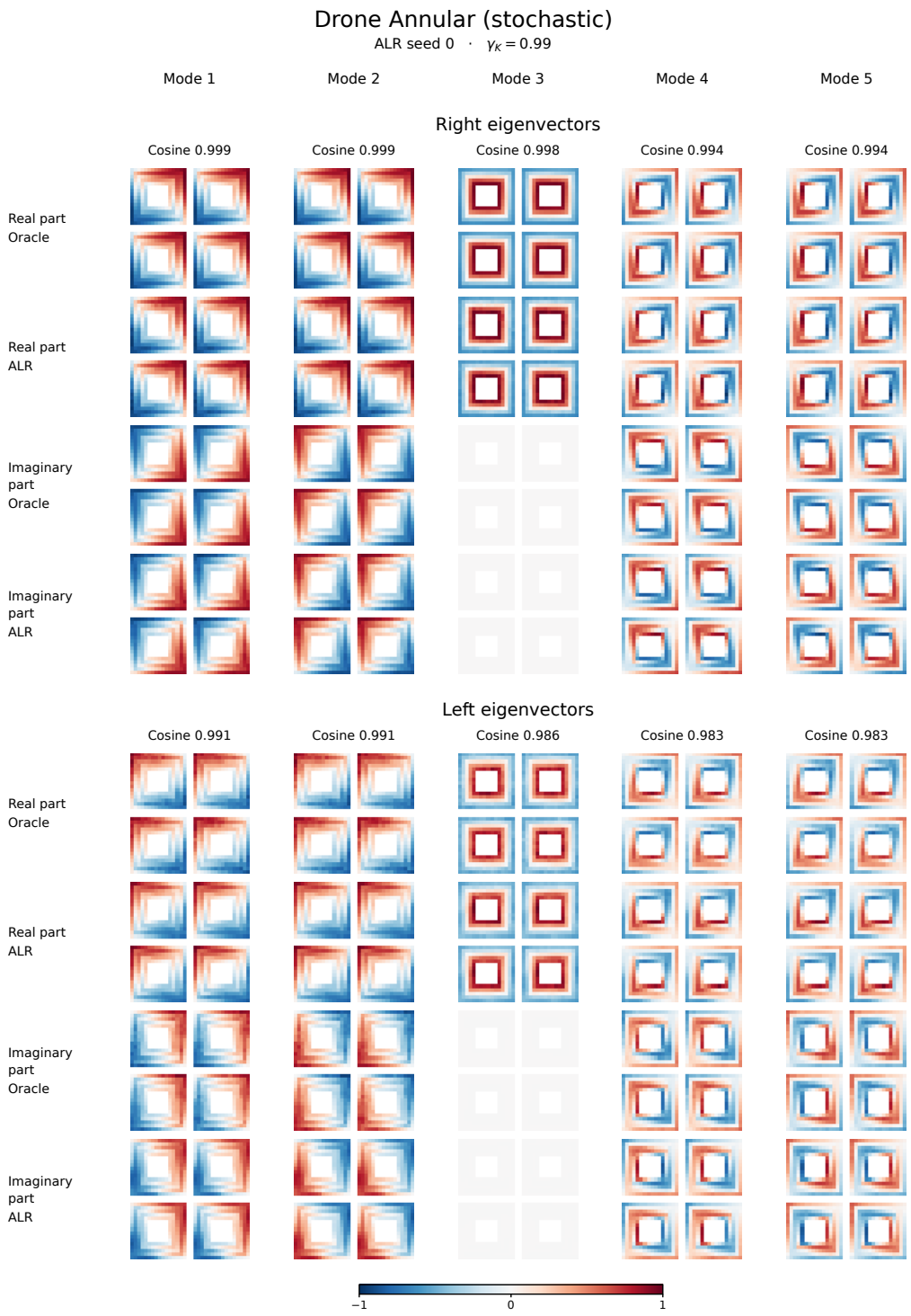
Drone Annular (stochastic)
ALR seed 0 · $\gamma_K = 0.99$
Mode 1
Mode 2
Mode 3
Mode 4
Mode 5
Right eigenvectors
Cosine 0.999
Cosine 0.999
Cosine 0.998
Cosine 0.994
Cosine 0.994
Real part Oracle
Real part ALR
Imaginary part Oracle
Imaginary part ALR
Left eigenvectors
Cosine 0.991
Cosine 0.991
Cosine 0.986
Cosine 0.983
Cosine 0.983
Real part Oracle
Real part ALR
Imaginary part Oracle
Imaginary part ALR
−1
0
1

Figure 10

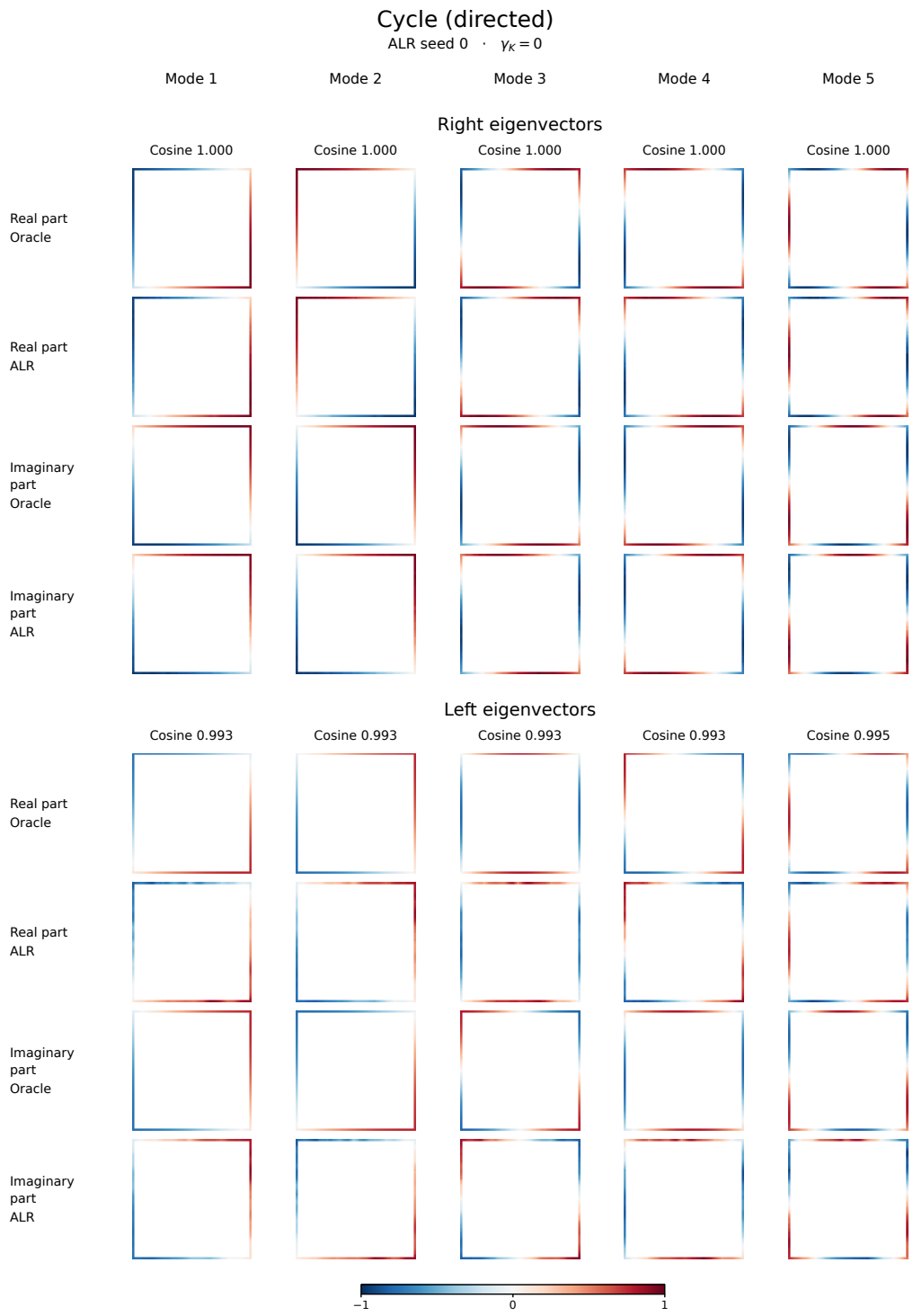
Cycle (directed)
ALR seed 0 · $\gamma_K = 0$
Mode 1
Mode 2
Mode 3
Mode 4
Mode 5
Right eigenvectors
Cosine 1.000
Cosine 1.000
Cosine 1.000
Cosine 1.000
Cosine 1.000
Real part Oracle
Real part ALR
Imaginary part Oracle
Imaginary part ALR
Left eigenvectors
Cosine 0.993
Cosine 0.993
Cosine 0.993
Cosine 0.993
Cosine 0.995
Real part Oracle
Real part ALR
Imaginary part Oracle
Imaginary part ALR
−1
0
1

Figure 11

C Stationary distribution comparison plots



Figure 12: Stationary distribution comparison plots. N is the number of state, π is the oracle stationary distribution and $\widehat{\pi}$ is the estimated stationary distribution taken as the left eigenvector associated with the lowest magnitude eigenvalue.

D DRONE DOMAIN VISUALIZATION

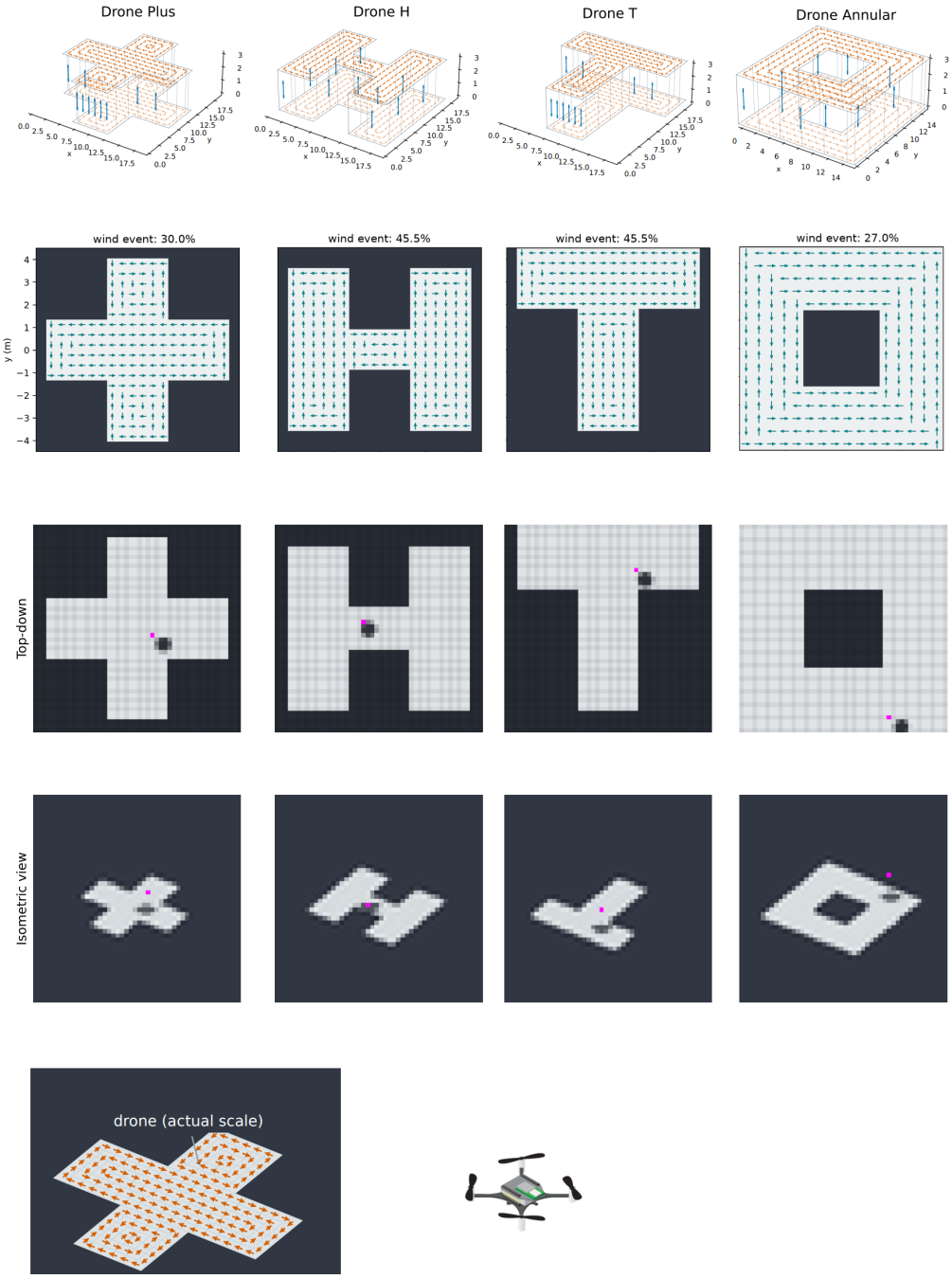

Figure 13

D.1 PARAMETERS

Learning for each method is performed with 16 random seeds per domain and observation type. We keep a batch size of 1024 for all methods, ALR and ALLO, across domains and state observation types where each method is evaluated. For training iterations with ALR, we do 100,000 iterations of Phase 1, where we learn the subspace vectors F_θ with learning rate 0.001 for numerical observations and 0.0003 for image observations. During phase 1 in ALR, trajectory samples use the standard multi-step sampling from GDO with discount factor 0.9 (Wu et al., 2019). For phase 2 in ALR, we do 50,000 iterations, where a set of K_h matrices is learnt with learning rate 0.0001 while keeping F_θ frozen. Neural networks output 50 vectors for

ALR (we hard-code the constant vector separately) and 51 vectors for ALLO. An Adam optimizer is used for all learning. ALLO uses 150,000 joint training iterations, a learning rate of 0.001 and a sampling discount of 0.9. fully connected hidden layers of 256, 256, 256 with a ReLU activation function for each layer. In image domains, we use a convolutional network prior to the MLP network with three layers, each with 16 channels, kernel 3×3, stride 2, padding $(1, 1)$ and ReLU. As with ALLO, all fully connected network weights are initialized using Haiku's default Linear-layer initialization. The convolutional layers instead use VarianceScaling(2, fan_in, truncated_normal). The input shape for the top-down image drone domains is $48 \times 48 \times 3$.

Table 1: Full learning and network parameters.

Parameter	**ALR**	**ALLO**	**Image ALR**
Random seeds	0–15	0–15	0–15
Batch size	1,024	1,024	1,024
Network outputs (d)	50 learnt	51 learnt	50 learnt
Constant mode	Hard-coded separately	Included in network outputs	Hard-coded separately
Evaluation dimensions	5, 10, 20, 50	5, 10, 20, 50	5, 10, 20, 50
Transitions per setting	Grids: 1,000,000; drones: 327,680	Grids: 1,000,000; drones: 327,680	327,680
Total training iterations	100,000 F + 50,000 K_h	150,000 joint	100,000 F + 50,000 K_h
F/encoder learning rate	10^{-3}	10^{-3}	3×10^{-4}
Phase-1 K learning rate	10^{-3}	—	3×10^{-4}
Phase-2 K_h learning rate	10^{-4}	—	10^{-4}
Training-pair discount	0.9 in Phase 1	0.9	0.9 in Phase 1
Orthogonality/norm coefficient	1	Adaptive barrier and duals	1
Horizons	Grids: $\{1, 4, 8, 16, 32, 50\}$; drones: $\{1, 4, 8, 16, 32, 64\}$	—	$\{1, 4, 8, 16, 32, 64\}$
Initial K in Phase 1	Identity	—	Identity
K_h initialization in Phase 2	All reset to identity	—	All reset to identity
Phase-2 matrices	Six forward matrices, each 50×50	—	Six forward matrices, each 50×50
-1 target	Current $F(Y)$, stop-gradient	Smoothness updates both $F(X)$ and $F(Y)$	Current $F(Y)$, stop-gradient
Phase-2 representation	Both $F(X)$ and $F(Y)$ frozen	—	Both $F(X)$ and $F(Y)$ frozen
Fully connected hidden layers	256, 256, 256; ReLU	256, 256, 256; ReLU	256, 256, 256; ReLU
CNN before MLP	None	None	Three layers; each has 16 output channels, 3×3 kernel, stride 2, padding $(1, 1)$ and ReLU
Fully connected initialization	Haiku Linear default; zero bias	Haiku Linear default; zero bias	Haiku Linear default; zero bias
Convolutional initialization	—	—	`VarianceScaling`: scale 2; `fan_in`; `truncated_normal`
Optimization			
Optimizer	Adam per encoder / per K	Adam for primal parameters	Adam per encoder / per K

Table 1: Optimization parameters.

Parameter	ALR	ALLO	Image ALR
Adam settings	$\beta_1 = 0.9, \beta_2 = 0.999,$ $\epsilon = 10^{-8}, \epsilon_{\text{root}} = 0$	$\beta_1 = 0.9, \beta_2 = 0.999,$ $\epsilon = 10^{-8}, \epsilon_{\text{root}} = 0$	$\beta_1 = 0.9, \beta_2 = 0.999,$ $\epsilon = 10^{-8}, \epsilon_{\text{root}} = 0$
ALLO barrier and dual configuration			
Initial barrier	—	0.5	—
Barrier coefficient rate	—	10^{-2}	—
Dual learning rate	—	10^{-4}	—
Dual-velocity rate	—	10^{-1}	—

E AI DISCLOSURE SECTION

In this work, we used generative AI to find related work. We have not used generative AI tools for research ideation or, generating synthetic datasets, proving mathematical claims, drafting sections of the paper, polishing writing, or for the implementation of the methods in this paper.