Trajectory Covering Features for Systematic Exploration in POMDP

Kaicheng Guo
Affiliation
Address
email@domain.edu

Abstract

Exploration in partially observable domains remain an open problem. Exploration in partially observable environments must operates over representations of histories rather than states, but learning a usefull history representation requires the trajectory coverge that exploration is meant to provide. We break this circular dependency with random recurrent featurizations: fixed, randomely-initialized recurrences that compresses histories into stationary fixed-size embeddings. We introduce three recurrent featurizations in this paper: a frozen GRU, a product of random Householder reflections, and a block-diagonal rotation. Across four partially observable environments and two frameworks, all recurrent methods outperforms the non-recurrent baselines.

1 Introduction

Partial observability adds subtantial complexity to exploration in reinforcement learning. With Markovian states, techniques like density estimation are commonlly used to approximate an agent's state visitation and produce bonuses that incentize exploration [Bellemare et al., 2016, Burda et al., 2018, Lobel et al., 2023]. Without access to the ground-truth state, however, the object of exploration becomes less clear: over what space is an agent meant to explore?

One natural approach is for an agent to explore over its trajectores. The difficulty is that the space of trajectories blows up exponentially with time, since trajectories grow with the horizon. Recurrent functions are a useful tool to compress histories into fixed-size representations and avoid this growth in computational complexitiy. In modern machine learning, recurrent functions such as recurrent neural networks are typicallly *learned* functions of trajectories, which embeds structure over the hidden space. But learning comes with a cost: learned representations changes over time, and calculating visitations over a non-stationary representation give a poor approximation of novelty. A history that was novel under an earlier representation may map to the same code as a familiar one after the representation drifts, and vice versa, so the resulting visitation counts no longer reflect what the agent has actually seen.

One option for a fixed-length and stationary history compression is a fixed random recurrent function. While random recurrences are a poor proxy for the ground-truth state, our goal is not to recover the state or the belief but to estimate statistics of state visitation, which is a coarser measure. Random functions, specifically predicting the output of fixed random neural networks, have been proposed as a way of aproximating density estimation for state visitation [Burda et al., 2018], but prior approaches have never considred random *recurrences* as a basis for exploration.

This work investigate density estimation with random recurrences for exploration in partially observable environments. We make the folowing contributions:

(1) We extend the standrad count-based exploration framework to the partially observable setting by replacing state counts with counts over a fixed featurization of histories, and articulate the resulting circular dependancy between exploration and history representation.

Preprint.

(2) We introduce the Lock environment, a small partially observable testbed in which reward depends on a hidden modular condition over the entire trajectory of observations, isolating the need to integrate information across an episode.

(3) We propose three random recurrent featurizations as fixed history compressions for exploration: a randomly initialized GRU, a recurrence built from a product of random Householder reflections, and a block-diagonal rotation whose blocks rotate at logarithmically spaced frequencies. The two orthogonal constructions are norm-preserving and retain information about distant prefixes that the random GRU contracts away.

(4) We empiricallly evaluate these featurizations as the target network in Random Network Distillation and as the front-end of a Coin Flip Network across four partially observable environments. The orthogonal constructions consistently match or outperform the random GRU and non-recurrent baselines, with the largest gap on long-horizon tasks (Section 6).

2 Background

We consider the problem setting of Markov decision proceses (MDPs) and its partially observable analog. An MDP is defined by its state space $\mathcal{S}$, action space $\mathcal{A}$, reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, state transition function $T : \mathcal{S} \times \mathcal{A} \to \Delta\mathcal{S}$, initial state distribution $p_0 \in \Delta\mathcal{S}$ and discount factor $\gamma \in [0, 1]$. States have the property that they are Markov, where the state and action are a sufficient statistic to predict the next state and reward $P(s_{t+1}, r_t \mid s_t, a_t) = P(s_{t+1}, r_t \mid s_t, a_t, \ldots, s_0, a_0)$. A partially observable Markov decision process (POMDP) extends MDPs by defining an observation function $\Phi : \mathcal{S} \to \Delta\mathcal{O}$. In general, observations $o \in \mathcal{O}$ are not Markov, and a statisic of the past observations and actions is required for future predictions. We define sequences of observation-action pairs as the space of all possible trajectories $\mathcal{T} = (\mathcal{O} \times \mathcal{A})^* = \bigcup_{n=0}^{\infty} (\mathcal{O} \times \mathcal{A})^t$. As a shorthand, we denote an observation-action pair at time t as $x_t = (o_t, a_t)$ with $x_t \in \mathcal{X} = \mathcal{O} \times \mathcal{A}$. In the POMDP setting, an agent's goal is to find a policy that maps the observed trajectory at time t, $\tau_t \in \mathcal{T}$, to a next action $\pi : \mathcal{T} \to \Delta\mathcal{A}$ to maximize its discounted future return $g_t = \sum_{i=0}^{\infty} \gamma^i r_i$, where $r_i = R(s_i, a_i)$ with $s_i \in \mathcal{S}$, $a_i \in \mathcal{A}$. The expectation of which is the value function $V_\pi(s) = \mathbb{E}_\pi[g_t \mid s_t = s]$.

A recurrent function is a function over sequences with a recursive state update. If the recursive state and input lie in the sets $\mathcal{H}$ and $\mathcal{X}$ respectively, then the recurrent function F is defined as $F : \mathcal{H} \times \mathcal{X} \to \mathcal{H}$, mapping the previous state and current input to a new state. Given an initial state $h_0 \in \mathcal{H}$ and a trajectory $\tau = (x_1, \ldots, x_T) \in \mathcal{X}^T$, applying F recursivly produces the state sequence $h_t = F(h_{t-1}, x_t)$ for $t = 1, \ldots, T$. We define the memory function $m_T : \mathcal{T} \to \mathcal{H}$ as the output of the recurrent state at the end of the trajectory, $m_T(\tau) \doteq h_T$, which compresses a variable-length history into a fix-sized representation in $\mathcal{H}$. The remainder of the paper studies different choices of F and the resulting memory functions m_T as compact featurizations of trajectories for exploration.

3 Exploration in Partially Observable Environments

Exploration in POMDPs is more comlex than in the fully observable setting. In MDPs, exploration serves a singel purpose: ensuring sufficent coverage of the state-action space during training. In POMDPs, an agent has no access of state, and instead must explore over its experienced trajectories.

3.1 From State Counts to Trajectory Counts

In fully observable environments, when the number of states is small, an agent can simply count how often each state-action pair has been visited. Using this count as an exploration bonus, an agent can learn a near-optimal policy [Strehl and Littman, 2008]. When the state space is large, the notion of visitation counts has been generalized to pseudocounts [Bellemare et al., 2016] and a variety of methods have been proposed to approximate them [Burda et al., 2018, Lobel et al., 2023].

$$N(s, a) \;\longrightarrow\; \text{bonus} = f\big(N(s, a)\big) \;\longrightarrow\; \text{coverage of state-action space.} \tag{1}$$

In partially observable environments, the agents no longer observe state s_t. Analogous to state counts, an agent must maintain a count over *histories*:

$$N\big(m_T(\tau)\big) \;\longrightarrow\; \text{bonus} = f\big(N(m_T(\tau))\big) \;\longrightarrow\; \text{coverage of trajectory space.} \tag{2}$$

Here $\tau = (o_{0:T}, a_{0:T-1})$ denotes a trajectory and $m_T(\tau)$ is an encoding of the trajectory. Encoding a trajectory in an efficient manner can be a difficult task, especially since the space of trajectories grows exponentially with time.

3.2 Lock

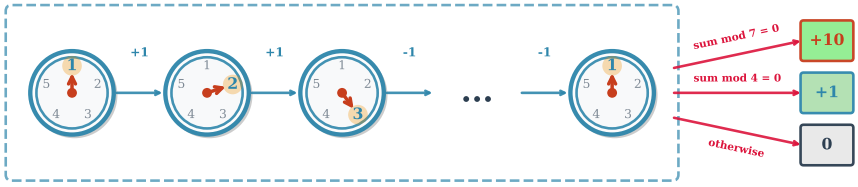


Figure 1: The Lock environment. The agent interacts with a circular lock with states $\{1, 2, 3, 4, 5\}$. Each action increments or decrements the number by 1. With probability 20%, the next state is resampled uniformly from $\{1, 2, 3, 4, 5\}$ instead of following the action. The initial number is drawn uniformly from $\{1, 2, 3, 4, 5\}$. The observation is the current number. At the end of a 20-step episode, the agent receives reward $+1$ if $\sum_{t=1}^{20} o_t \equiv 0 \pmod 4$, and reward $+10$ if $\sum_{t=1}^{20} o_t \equiv 0 \pmod 7$.

To better understand what makes exploration in partially observable environments fundamentally different, we introduce the Lock environment in Figure 1. In this environment, the agent navigates a 5-state circular lock for 20 steps, observing only the current number. Rewards depend only on whether the sum of the entire trajectory of observations satisfies a hidden modular condition. With some probability, the next state is randomly reset, introducing additional stochasticity. Importantly, the observation at any single timestep provides no direct signal of whether the cumulative sequence will satisfy the modular condition. As a result, exploration based solely on immediate observations is insufficient. To succeed, the agent must reason over entire observation action sequences and account for stochasticity in order to uncover the underlying modular reward structure.

3.3 Circular Dependencies

Exploring directly over raw trajectories is computationally intractable, since the number of possible histories grows exponentially with time. We therefore need a compact featurization $m_T(\tau)$ that compresses a history into a low-dimensional embedding while preserving the relevant information relevant. The ideal choice is an optimal memory function: a map from histories to a sufficient statistic of the belief over latent states, so that two trajectories share a code precisely when they are equivalent under the optimal policy. With such a featurization, the trajectory counts in Eq. 2 would play the same role for POMDPs that state counts play for MDPs.

This sets up a circular dependency. Learning a useful memory function requires trajectory data that broadly covers the space of histories, which is precisely what exploration is meant to provide. But directed exploration in turn requires a featurization to count over: without one, every raw trajectory looks unique and visitation counts carry no signal. Exploration depends on the memory function, and the memory function depends on exploration. Resolving this circularity is the central methodological challenge that motivates the rest of the paper.

4 Random Recurrent methods

One solution we have for the circular dependencies is random memory functions. Instead of using learned memory functions, we use randomly intialized memory function. In the following subsections, we will present different random recurrent sturcture that we can use for exploration that can successfully separate trajectories.

4.1 Random GRU

The simplest random recurrent featurization is a Gated Recurrent Unit (GRU) [Cho et al., 2014] with frozen, randomly initialized parameters. Let ϕ be a fixed observation encoder and ψ the GRU parameters, both sampled at initialization and held constant throughout training. We define $m_T(\tau)$ as

the hidden state of the recurrence

$$e_t = \phi(o_t, a_{t-1}), \tag{3}$$
$$h_t = \mathrm{GRU}_\psi(h_{t-1}, e_t), \tag{4}$$
$$m_T(\tau) \doteq h_T. \tag{5}$$

Because ψ is random and the recurrence is nonlinear, two trajectories with distinct prefixes are mapped to distinct hidden states with high probability. However, a randomly initialized GRU is not norm-preserving: its state-to-state Jacobian contracts under repeated application, so distant prefixes of τ may get washed out of h_T. In the following sections, we will introduce two norm-preserving memory structure whose state transition is orthogonal.

4.2 Reflection Matrix

The simplest such construction is a product of Householder reflections. Given a unit vector $u \in \mathbb{R}^d$, the elementary reflector about the hyperplane $u^\perp$ is

$$R(u) \doteq I - 2uu^\top, \qquad \|u\| = 1. \tag{6}$$

$R(u)$ is symmetric, orthogonal, and involutive ($R(u)^2 = I$), and it preserves Euclidean norm exactly: $\|R(u)h\| = \|h\|$ for all h. We sample N unit vectors $u_1, \ldots, u_N$ uniformly from the sphere S^{d-1} at initialization, freeze them, and define the random reflection operator

$$R \doteq R(u_N)\, R(u_{N-1}) \cdots R(u_1). \tag{7}$$

R is itself orthogonal, and any orthogonal matrix in $\mathbb{R}^{d\times d}$ can be written as a product of at most d such reflections [Artin, 1957], so the family is fully expressive in the limit $N \to d$.

We use R as the recurrence kernel of the memory function:

$$e_t = \phi(o_t, a_{t-1}), \tag{8}$$
$$h_t = R\, h_{t-1} + W e_t, \tag{9}$$
$$m_T(\tau) \doteq h_T, \tag{10}$$

where ϕ, W, and $\{u_i\}_{i=1}^N$ are sampled at initialization and held frozen. Because R is orthogonal, $\partial h_t / \partial h_{t-1}$ has unit singular values and the recurrence neither contracts nor explodes: information injected at any timestep is preserved in h_T up to the geometry imposed by subsequent inputs. In Lock, this means that two histories differing only in their first observation remain distinguishable in h_T regardless of T, which is precisely the property the GRU lacked.

4.3 Rotation Matrix

Reflections are orthogonal but unstructured: the same matrix is applied at every timestep, so the operator carries no notion of when an event occurred along the trajectory. An alternative construction makes time explicit by using a block-diagonal rotation whose blocks rotate at distinct frequencies.

For an even hidden dimension d, partition the state $h \in \mathbb{R}^d$ into $d/2$ coordinate pairs, and assign each pair a rotation angle θ_i. Define the elementary 2×2 rotation

$$R_2(\theta) \doteq \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}, \tag{11}$$

and the block-diagonal rotation operator

$$R(\boldsymbol{\theta}) \doteq R_2(\theta_1) \oplus R_2(\theta_2) \oplus \cdots \oplus R_2(\theta_{d/2}). \tag{12}$$

$R(\boldsymbol{\theta})$ is orthogonal, $\det R(\boldsymbol{\theta}) = 1$, and $\|R(\boldsymbol{\theta})h\| = \|h\|$ for all h.

Rather than sampling $\boldsymbol{\theta}$ uniformly, we set the angles by their periods. Let $T_{\min}$ and $T_{\max}$ denote the shortest and longest periods we wish to resolve. We log-space the periods across the $d/2$ blocks,

$$T_i \doteq T_{\min} \left(\frac{T_{\max}}{T_{\min}}\right)^{(i-1)/(d/2-1)}, \qquad \theta_i \doteq \frac{2\pi}{T_i}, \tag{13}$$

so block i completes one full revolution every T_i recurrence steps. The fastest block ($T_1 = T_{\min}$) distinguishes trajectories that differ at consecutive steps, while the slowest block ($T_{d/2} = T_{\max}$)

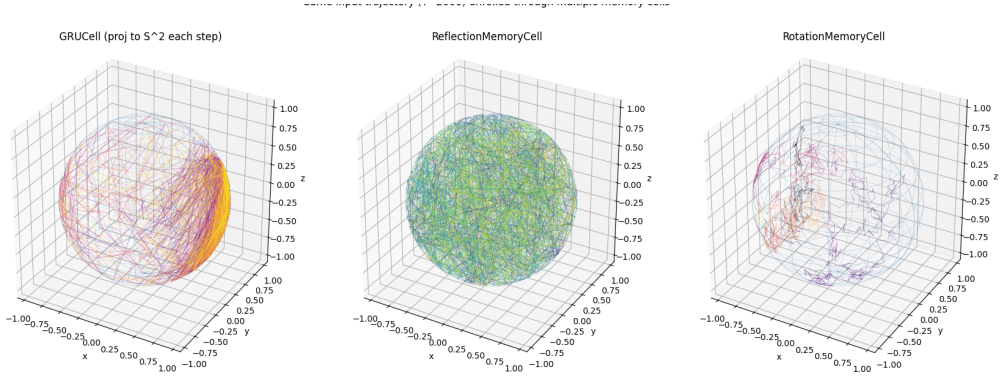


Figure 2: Hidden-state orbits on S^2 for the three memory cells driven by the same i.i.d. Gaussian input trajectory of length $T = 2000$. Left: random GRU (with per-step projection onto S^2). Middle: reflection cell. Right: rotation cell. The random GRU concentrates on an attractor; the reflection cell covers the sphere approximately uniformly; the rotation cell traces a structured orbit reflecting its periodic kernel.

distinguishes trajectories that differ over the whole episode. In our experiments we set $T_{\min} = 1$ and $T_{\max}$ proportional to the maximum episode length.

The memory function uses $R(\boldsymbol{\theta})$ as the recurrence kernel:

$$e_t = \phi(o_t, a_{t-1}), \tag{14}$$

$$h_t = R(\boldsymbol{\theta})\, h_{t-1} + W e_t, \tag{15}$$

$$m_T(\tau) \doteq h_T, \tag{16}$$

where ϕ, W, and the period schedule $\{T_i\}_{i=1}^{d/2}$ are fixed at initialization. Unrolling the recurrence yields

$$h_T = R(\boldsymbol{\theta})^T h_0 + \sum_{t=1}^{T} R(\boldsymbol{\theta})^{T-t}\, W e_t, \tag{17}$$

and because $R(\boldsymbol{\theta})^k = R(k\boldsymbol{\theta})$ acts on block i by rotating it through angle $k\theta_i = 2\pi k/T_i$, the contribution of input e_t is encoded in h_T by its relative position $T - t$ within the trajectory. Two histories that agree on the multiset of inputs but differ in their order produce different h_T, and two histories that agree on order but differ in any single e_t are distinguished by every block whose period does not divide $T - t$.

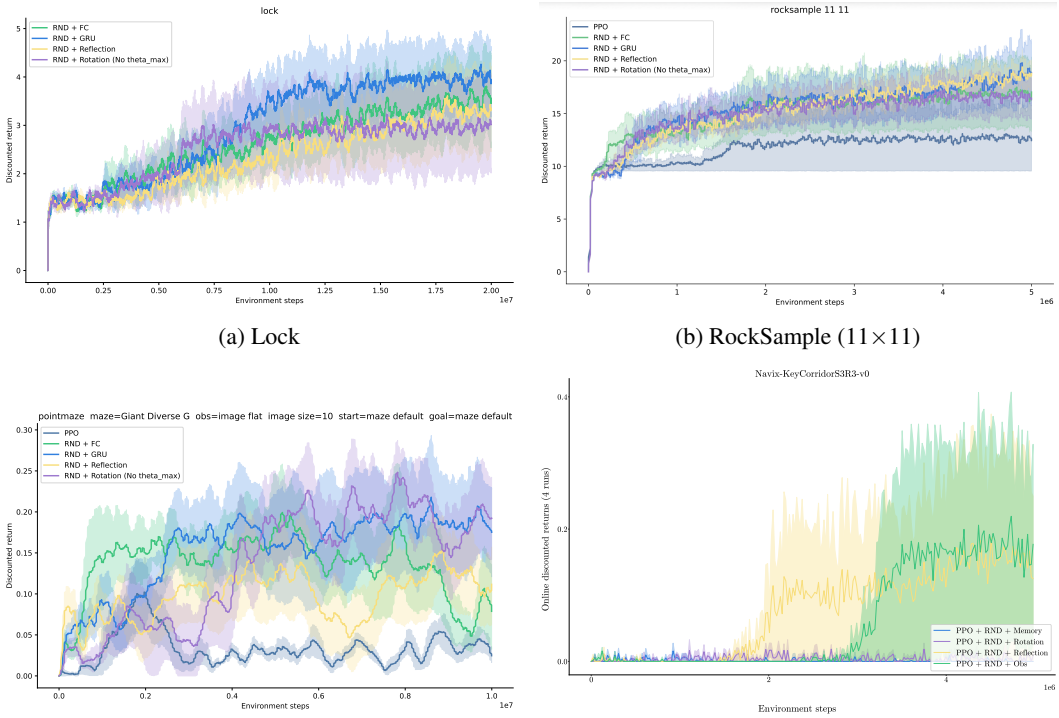
Compared to the reflection construction of Section 4.2, the rotation operator trades expressivity for structure: a single shared rotation cannot realize an arbitrary element of $O(d)$, but it embeds positional information directly into the recurrence and exposes a small number of interpretable hyperparameters ($T_{\min}$, $T_{\max}$) in place of N random unit vectors.

5 Memory Orbit Geometry

Before integrating the random recurrent featurizations into a full reinforcement learning pipeline, we run a diagnostic experiment that examines their hidden-state geometry on a synthetic input stream. The goal is to verify that each cell produces the qualitative behavior its construction promises: a contractive orbit for the random GRU, a uniformly distributed orbit for the reflection cell, and a structured, low-dimensional orbit for the rotation cell.

We construct a fixed input trajectory $\tau = (e_1, \ldots, e_T)$ of length $T = 2000$, with each e_t drawn i.i.d. from a standard Gaussian and projected onto the unit sphere. We feed τ into each of the three memory cells with hidden dimension $d = 3$ and visualize the orbit $\{h_t\}_{t=1}^{T}$ on S^2. For the random GRU, which is not norm-preserving, we additionally project h_t onto the unit sphere at each step so that all three orbits live in the same ambient space. Figure 2 shows the resulting trajectories.

The three cells produce visibly different orbits. The random GRU concentrates on a small region of the sphere: even with explicit per-step normalization, the underlying recurrence is contractive and



(a) Lock

(b) RockSample (11×11)

(c) PointMaze

(d) KeyCorridor

Figure 3: PPO + RND with different recurrent target networks on the four evaluation environments. Curves show discounted return averaged across 4–5 seeds with shaded one-standard-deviation regions. Reflection and rotation targets consistently outperform the GRU target and the non-recurrent baselines, with the gap most pronounced on the long-horizon Lock and KeyCorridor tasks.

the orbit is dragged toward an attractor determined by the random parameters. The reflection cell covers S^2 approximately uniformly, consistent with the fact that a product of random Householder reflections is an unstructured random orthogonal operator. The rotation cell traces a more structured orbit whose support reflects the periodic structure of its block-diagonal kernel, with each frequency block contributing a distinguishable temporal signature. These observations confirm the analysis of Section 3: the random GRU loses information about distant prefixes, while the two orthogonal constructions preserve it, with the rotation cell additionally encoding position.

6 Experiments

We evaluate the three random recurrent featurizations of Section 3 on four partially observable reinforcement learning environments that span a range of observation modalities, episode lengths, and exploration difficulties.

(1) **Lock** (Section 3.2): a 5-state circular lock in which reward depends on a hidden modular condition over the sum of all 20 observations, isolating the need to integrate information across an entire trajectory.

(2) **RockSample** [Smith and Simmons, 2012]: a classic POMDP in which the agent navigates a grid containing rocks of unknown quality and receives noisy sensor readings whose accuracy decays with distance, requiring the agent to move closer to disambiguate good rocks from bad ones before sampling.

(3) **PointMaze** [Radji, 2025]: a continuous-control navigation task in which a point-mass agent must reach a goal location in a maze from egocentric observations, so the agent never directly observes either the global maze topology or the goal position.

(4) **KeyCorridor** [Chevalier-Boisvert et al., 2023]: a MiniGrid task in which the agent must locate a key in one of several rooms and use it to unlock a target room containing a goal object, combining sparse rewards with long-horizon dependencies between earlier and later observations.

6.1 Exploration via Random Network Distillation

We first instantiate the random recurrent featurizations as the target network in Random Network Distillation (RND) [Burda et al., 2018]. In the fully observable setting, RND assigns an exploration bonus equal to the squared error between a learned predictor f and a frozen random target g on the current state s_t:

$$b(s_t) \;=\; \|f(s_t) - g(s_t)\|^2. \tag{18}$$

The predictor is trained to minimize this error on observed states, so $b(s_t)$ decays as a state is visited more often and serves as a soft visitation count.

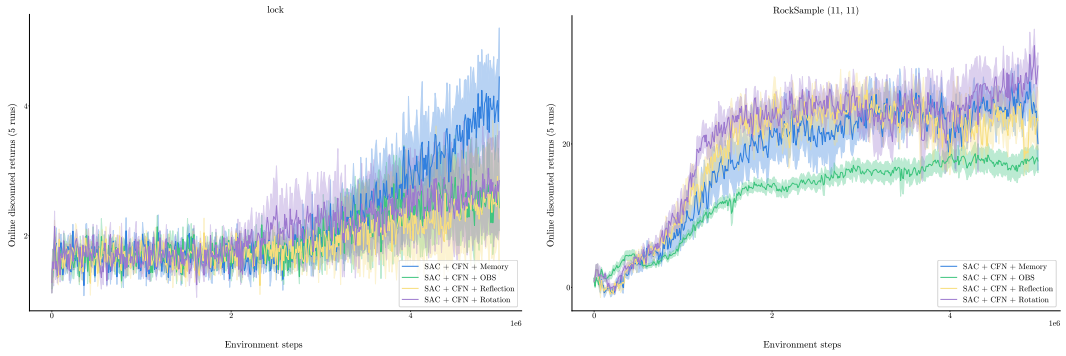
To adapt RND to the POMDP setting, we make two changes. First, the input to both networks becomes the observation o_t rather than the latent state s_t. Second, both networks become recurrent: the trainable predictor f is a standard learned GRU, and the frozen target g is one of the three random recurrent featurizations from Section 3 (random GRU, reflection, or rotation). The exploration bonus becomes

$$b(\tau_{:t}) \;=\; \left\|f_\theta(\tau_{:t}) - g(\tau_{:t})\right\|^2, \tag{19}$$

so that the bonus on a history is small only when the predictor has seen sufficiently many similar histories under the target featurization. We use PPO [Schulman et al., 2017] as the base agent and add the bonus to the environment reward.

We compare our recurent methods against RND with a non-recurrent fully connected target on observations (RND + FC/OBS), and on PointMaze additionally against vanilla PPO without an intrinsic reward bonus. Figure 3 reports discounted return as a function of environment steps, averaged across 5 seeds with shaded one-standard-deviation regions.

On Lock, the random GRU target outperforms all other variants. On RockSample, the reflection and GRU targets outperform the rotation and FC baselines. On PointMaze, all three recurrent targets (GRU, reflection, rotation) outperform RND + FC. On KeyCorridor, the reflection target achieves the highest return. Across the four environments, every recurrent variant outperforms the non-recurrent RND + FC baseline on at least three tasks, while no single recurrent featurization dominates on all four.



(a) Lock (b) RockSample (11×11)

Figure 4: SAC + CFN with different front-ends on Lock and RockSample (11×11). Curves show online discounted return over five seeds with shaded one-standard-deviation regions. The recurrent featurizations outperform the observation-only baseline, with reflection and rotation matching or exceeding the random GRU.

6.2 Exploration via Coin Flip Networks

To check that the conclusions of Section 6.1 are not specific to RND, we repeat the experiment with Coin Flip Networks (CFN) [Lobel et al., 2023]. CFN assigns each state a random Rademacher vector $c \in \{-1, +1\}^k$ at each visit and trains a network z_θ to predict the average coin-flip vector. The

squared norm $\|z_\theta(s)\|^2$ then approximates the count-based bonus $1/\sqrt{N(s)}$, providing a more direct pseudocount estimator than RND.

We adapt CFN to POMDPs analogously: z_θ takes the observation history rather than the state, combined with one of the three random recurrent featurizations to obtain a trajectory-level pseudocount. We use SAC [Haarnoja et al., 2018] as the base agent and add the bonus to the environment reward. We compare four variants on Lock and RockSample: SAC + CFN with each of the three recurrent featurizations as the front-end (GRU, Reflection, Rotation), and SAC + CFN + OBS, a non-recurrent baseline that uses raw observations.

Figure 4 shows the results. On Lock, the random GRU achieves the highest final return, with the reflection and rotation variants reaching intermediate performance above the OBS baseline. On RockSample, where the single observation signal is strong enough that the OBS baseline learns nontrivial behavior, the three recurrent variants nonetheless converge faster and to a higher discounted return.

6.3 Discussion

Two findings emerge from these experiments. First, replacing observation-level exploration bonuses with their trajectory-level analogs is necessary and sufficient on every environment we tested: the non-recurrent RND + FC and CFN + OBS baselines fail on the long-horizon tasks (Lock, KeyCorridor) and underperform on the others, while every recurrent variant clears the exploration bottleneck. This is consistent with the argument in Section 3.1 that count-based exploration in POMDPs must operate over histories rather than observations.

Second, the choice of random recurrent featurization matters, but no single recurrence dominates across environments. The random GRU achieves the highest return on Lock under both RND and CFN. Reflection wins on KeyCorridor, GRU and reflection match each other on RockSample, and all three recurrent variants are closely matched on PointMaze. This pattern suggests that the interaction between a recurrence's geometry and the task's reward structure, observation distribution, and horizon appears to matter at least as much as the geometry itself, and characterizing that interaction is an open question we leave to future work.

7 Conclusion

We studied count-based exploration in partially observable environments, where the absence of Markovian states forces exploration to operate over histories rather than observations. We argued that this creates a circular dependency between exploration and history representation: learning a useful memory function requires the trajectory coverage that exploration is meant to provide and we proposed random recurrent featurizations as a way to break the cycle. By fixing the recurrence at initialization, we obtain a stationary featurization over which visitation counts are well-defined.

We introduced three constructions: a frozen GRU, a product of random Householder reflections, and a block-diagonal rotation with logarithmically-spaced periods and instantiated each as the target network in Random Network Distillation and as the front-end of a Coin Flip Network. Across four partially observable environments, every recurrent featurization improves over its non-recurrent counterpart, confirming that trajectory-level pseudocounts are the right object for exploration in POMDPs. Among the recurrent variants, no single recurrence dominates: the random GRU performs best on Lock under both pseudocount frameworks, reflection wins on KeyCorridor, and the three are closely matched on RockSample and PointMaze.

The orbit diagnostic of Section 5 shows that the three recurrences produce qualitatively distinct hidden-state geometries, but those geometric differences do not predict downstream exploration performance in the way our initial analysis suggested. We see this as the central open question raised by our experiments: when does norm preservation, attractor structure, or positional encoding actually matter for exploration, and what property of a task determines which recurrence is the right one? We also leave to future work the natural extension of these ideas beyond random recurrences, including learned-but-stationary featurizations and recurrences whose periods or reflection schedules adapt to the task at hand.

References

Emil Artin. Geometric Algebra. Interscience Publishers, New York, 1957.

Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation, 2016. URL https://arxiv.org/abs/1606.01868.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation, 2018. URL https://arxiv.org/abs/1810.12894.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. In Advances in Neural Information Processing Systems 36, New Orleans, LA, USA, December 2023.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014. URL https://arxiv.org/abs/1406.1078.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In International Conference on Machine Learning, pages 1861–1870. PMLR, 2018.

Sam Lobel, Akhil Bagaria, and George Konidaris. Flipping coins to estimate pseudocounts for exploration in reinforcement learning, 2023. URL https://arxiv.org/abs/2306.03186.

Waris Radji. Pointax: Jax-native pointmaze environment, 2025. URL https://github.com/riiswa/pointax.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.

Trey Smith and Reid Simmons. Heuristic search value iteration for pomdps, 2012. URL https://arxiv.org/abs/1207.4166.

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. ISSN 0022-0000. doi: https://doi.org/10.1016/j.jcss.2007.08.009. URL https://www.sciencedirect.com/science/article/pii/S0022000008000767. Learning Theory 2005.

A Technical appendices and supplementary material

Technical appendices with additional results, figures, graphs, and proofs may be submitted with the paper submission before the full submission deadline. There is no page limit for the technical appendices. The paper must be able to stand alone without the appendix.