A Geometric Lens on RL Environment Complexity Based on Ricci Curvature

Ali Saheb Pasand[1,2], **Pablo Samuel Castro**[2,3,4,†], **Pouya Bashivan**[1,2,†]
{ali.sahebpasand,pablo-samuel.castro,bashivap}@mila.quebec

[1]**McGill University, Canada**
[2]**Mila-Québec AI Institute**
[3]**Université de Montréal, Canada**
[4]**Google DeepMind**

[†] Contributed equally as co-senior authors

Abstract

We introduce Ollivier-Ricci Curvature (ORC) as an information-geometric tool for analyzing the local structure of reinforcement learning (RL) environments. We establish a novel connection between ORC and the Successor Representation (SR), enabling a geometric interpretation of environment dynamics decoupled from reward signals. Our analysis shows that states with positive and negative ORC values correspond to regions where random walks converge and diverge respectively, which are often critical for effective exploration. ORC is highly correlated with established environment complexity metrics, yet integrates naturally with standard RL frameworks based on SR and provides both global and local complexity measures. Leveraging this property, we propose an ORC-based intrinsic reward that guides agents toward divergent regions and away from convergent traps. Empirical results demonstrate that our curvature-driven reward substantially improves exploration performance across diverse environments, outperforming both random and count-based intrinsic baselines.

1 Introduction

Estimating and understanding the local and structural complexity of reinforcement learning (RL) environments is important for building learning algorithms that are both robust and sample efficient. While global properties like overall task difficulty (Laidlaw et al., 2023), benchmark performance (Aitchison et al., 2023), or reward sparsity (Ecoffet et al., 2019) are often studied, local complexity, which looks at how different parts of the environment vary in connectivity or transition dynamics, is less explored. In this work, we propose using Ollivier-Ricci Curvature (ORC) (Ollivier, 2009) as a well-established and interpretable measure of local structure in RL environments. ORC quantifies how random walks under a policy behave at different regions of space, revealing whether local trajectories tend to converge (positive curvature) or diverge (negative curvature). This provides a geometric and probabilistic perspective on environment structure that goes beyond simple state counts or visitation frequency. To apply ORC in reinforcement learning, we connect it to the Successor Representation (SR), a common method that separates environment dynamics from the reward. This connection allows us to compute ORC in a way that fits RL goals and can be used with existing SR-based methods. Next, we show that ORC is strongly related to well-known measures of environment complexity. Using this, we propose a new intrinsic reward based on curvature that encourages exploring divergent areas and avoids highly connected regions, leading to better exploration through more uniform and diverse state coverage.

2 Background

This section introduces Ollivier-Ricci curvature (Ricci & Levi-Civita, 1900) and the successor representation (Dayan, 1993), then presents a unified framework linking the two to compute curvature between states under a given policy.

2.1 Ollivier-Ricci Curvature (ORC)

Ricci curvature is a concept from information geometry that characterizes how different regions of a space contract or expand. Computing the exact Ricci curvature typically involves complex tensor-based calculations. As an alternative, Ollivier-Ricci Curvature (ORC) provides an approximation based on optimal transport theory and the probability distributions induced by random walks.

For two points x and y in a metric space, ORC compares the Wasserstein-1 distance (W_1) (Villani et al., 2008) between the **probability measures** centered at these points (μ_x and μ_y) with the **geodesic distance** $d(x, y)$. These probability measures describe how mass is distributed to neighboring points if we initiate random walks at x and y, respectively. for example, in graphs, μ_x can be defined as the uniform or weighted distribution over the neighbors of x. The curvature between x and y is defined as:

$$\kappa(x, y) = 1 - \frac{W_1(\mu_x, \mu_y)}{d(x, y)}. \tag{1}$$

This quantity measures how much closer (or farther) the local probability distributions are compared to the geodesic distance between the points. The following three cases may arise (illustrated in Figure 1):

- **Negative ORC**: Random walks originating from x and y tend to diverge (left image).
- **Zero ORC**: Random walks from x and y neither diverge nor converge significantly (center image).
- **Positive ORC**: Random walks from x and y tend to converge (right image).

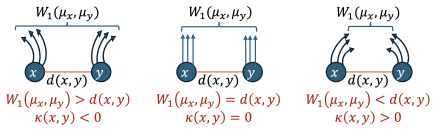

Figure 1: Illustration of ORC. Left: negative curvature where random walk distributions diverge. Center: zero curvature with neutral behavior. Right: positive curvature where distributions converge.

2.2 Successor Representation (SR)

The successor representation (SR) was first introduced by (Dayan, 1993) in the context of reinforcement learning as a method to disentangle the reward function from the environment dynamics. SR provides a notion of long-run neighborhoods under a given policy. In other words, instead of viewing states as isolated points in the state space, SR characterizes each state by the distribution of future states it is expected to visit. Mathematically, the SR between states s and s' is defined as:

$$\mathbf{SR}^{\pi}(s, s') = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \, \mathbb{I}\{s_t = s'\} \,\middle|\, s_0 = s\right] \tag{2}$$

This expression captures the expected discounted future occupancy of state s', starting from state s and following policy π. s is the starting state and γ is the discount factor.

2.3 Connecting SR and ORC

To compute ORC between two states in an RL environment, a probability measure must be defined at each state. A common approach is to construct a non-uniform distribution over immediate neighbors based on edge weights in the connectivity graph. However, this method, used by papers such as Ni et al. (2019), only considers the local neighborhood structures, which does not give information about the distribution induced on all other states in a long run. In this work, we define the probability measure at each state using the normalized rows of the SR matrix $\mathbf{SR}^\pi$, computed by approximating Equation 2. This reflects the distribution over future state occupancies induced by starting a random walk from state s. Specifically, in Equation 1, we define μ_s as:

$$\mu_s(s') = \frac{\mathbf{SR}^\pi(s, s')}{\sum_i \mathbf{SR}^\pi(s, s'_i)} \tag{3}$$

3 Methodology

In this section, we explain how we estimate SR and ORC in an offline way before starting the RL task. In Appendix C we explain how SR and ORC can be calculated in an online way as well.

3.1 SR Calculation

To estimate the successor representation (SR), we iterate over all states. For each state, we initiate a random walk of length L_{SR}. At each step t, we take an action based on the policy and increment the entry corresponding to the visited state by γ^t. This process is repeated N_{SR} times per starting state. Finally, we divide the accumulated values by N_{SR} to obtain the average. The resulting matrix has rows representing the discounted state visitation counts for random walks of length L_{SR} starting from each state. The pseudocode is shown in Appendix B Algorithm 1.

3.2 ORC Calculation

After computing the successor representation (SR) for all states, we estimate the Ollivier-Ricci curvature (ORC), which requires geodesic distances to compute both the Wasserstein distance (numerator) and the ground distance (denominator) in Equation 1. These distances are approximated by the shortest-path lengths on a connectivity graph over the state space. To construct this graph, we perform N_{ORC} random walks of length L_{ORC} starting from each state. At each step t, an action a is sampled from the policy $\pi(a \mid s)$, and the next state s' is reached. If there is no edge between s and s' in the adjacency matrix, a weight of t is assigned. If an edge already exists, the weight is updated to $\min(\text{current weight}, t)$. This process records the shortest observed step-count at which s' is reachable from s. After constructing the connectivity graph, we compute the Ollivier-Ricci curvature $\kappa^\pi(s, s')$ between each state s and its neighbors using Equation 1. The overall curvature at state s is given by the average of all pairwise curvatures $\kappa^\pi(s, \cdot)$. The pseudocode is shown in Appendix B Algorithm 2.

4 Experiments

In the following experiments, we aim to address the following research questions:

1. **RQ1: What are the geometric interpretations of positive and negative ORC values?**
2. **RQ2: Can statistical summaries of ORC values effectively measure the complexity of RL environments?**
3. **RQ3: Are ORC values useful as an intrinsic reward?**

To answer these questions, we use three types of environments: 1) Mazes with varying structure to study how ORC relates to topological complexity; 2) Rooms

with Bridges, where narrow connectors reveal ORC's ability to highlight bottlenecks; and 3) Tabular Atari, from Laidlaw et al. (2023), which includes transition/reward matrices and complexity metrics, enabling ORC analysis in discrete settings.

4.1 RQ1: What is the geometric interpretation of positive/negative Ricci curvature values?

As discussed in Section 2, different Ricci curvature values correspond to different types of regions:

1. **High positive Ricci** (Ricci $\gg 0$): Indicate tightly connected regions where nearby random walks converge, such as maze dead ends or room corners.
2. **Near-zero Ricci** (Ricci ≈ 0): Reflect flat or regular areas with neutral random walk behavior, like straight corridors or open spaces.
3. **Large negative Ricci** (Ricci $\ll 0$): Signal bottlenecks where walks diverge, e.g., maze junctions or narrow bridges.

We now empirically evaluate how well these interpretations align with computed ORC values across different environment regions.

4.1.1 Mazes and Rooms Connected with Bridges

To illustrate how ORC varies across maze regions, we compute Ricci curvature values in mazes with different Branching (B) and Winding (W) factors.

At each location, the agent can take one of three actions: move forward, turn left, or turn right, and face one of four orientations: up, right, down, or left. For simplicity, we aggregate all orientations at a location into a single state and compute Ricci curvature for this aggregated state. The Successor Representation (SR) and connectivity graph are derived using a random policy with uniform action probabilities.

Figure 2 shows ORC values across mazes with varying B and W (larger examples in Appendix E). As expected, dead ends and winding paths have high positive curvature (random walks stay local), straight corridors have near-zero curvature, and branching points show large negative curvature (walks diverge). Figure 3 shows ORC in room-based environments with bridges: corners typically exhibit high positive curvature, while bridges show strong negative values.

4.1.2 Tabular Atari

In Atari games, spatially mapping Ricci curvature is infeasible due to the high-dimensional, non-spatial nature of observations and the need for temporal context. To address this, we visualize sequences of five consecutive

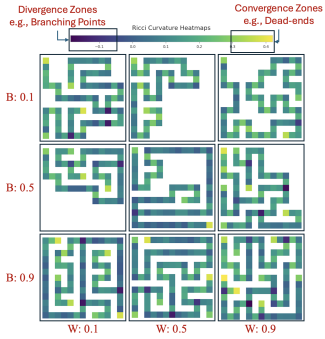

Figure 2: Ricci curvature values across different locations in mazes with varying **B**ranching and **W**inding factors. As observed, dead-ends (end of branches) and winding segments tend to exhibit large positive Ricci values, branching points correspond to large negative values, and straight corridors typically have Ricci values close to zero.

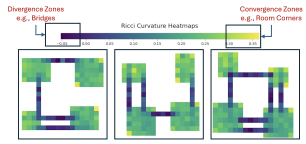

Figure 3: Ricci curvature values in rooms connected with bridges. As expected, corners of rooms exhibit high positive Ricci values, bridges show negative curvature, and the middle regions of rooms tend to have Ricci values close to zero.

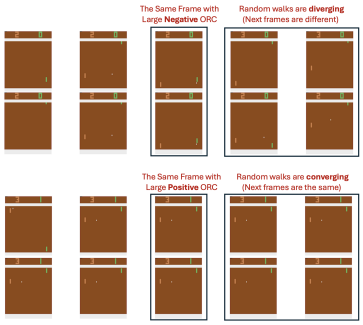

Figure 4: Sequences of five consecutive frames from Atari Pong. The top panel shows a frame with large negative ORC ($\kappa^{\pi} = -0.60$), leading to diverging futures. The bottom shows one with large positive ORC ($\kappa^{\pi} = 0.48$), leading to converging futures.

frames. Figure 4 shows four such sequences: in the top panel, the middle frame has large negative ORC, leading to diverging future frames; in the bottom, a large positive ORC leads to convergence. More examples are in Appendix F.

4.2 RQ2:Can statistical summaries of ORC values effectively measure the complexity of RL environments?

In this section, we first examine how various statistics of ORC values change as a function of three maze characteristics: (1) Size, (2) Branching Factor, and (3) Winding Factor. We then analyze the correlation between these Ricci curvature statistics and three measures of environment complexity, as introduced by Laidlaw et al. (2023): (1) *Effective Horizon* (higher values indicate greater complexity and shows long-horizon planning is needed), (2) *Probability of Finding the Optimal Reward*, and (3) *Minimum State-Action Occupancy*

4.2.1 Mazes

Room Size	(B, W)	Min	Max	Mean($\lvert ORC \rvert$)	STD	Range	Entropy Diff	α
15x15	(0.1, 0.1)	-0.04	0.35	0.11	0.06	0.40	0.18	42.29
	(0.5, 0.5)	-0.13	0.37	0.13	0.08	0.51	0.21	45.91
	(0.9, 0.9)	-0.17	0.38	0.14	0.10	0.55	0.24	64.14
21x21	(0.1, 0.1)	-0.06	0.37	0.14	0.08	0.44	0.38	59.57
	(0.5, 0.5)	-0.14	0.37	0.15	0.09	0.52	0.43	63.39
	(0.9, 0.9)	-0.18	0.40	0.17	0.10	0.57	0.45	67.76

Table 1: Change in Ricci Curvature's statistics by changing the maze's characteristics. Values are averaged over 10 mazes.

Table 1 summarizes Ricci curvature statistics for mazes varying in **branching** (B), **winding** (W), and **size**. Each row represents a (B, W) pair in 15×15 and 21×21 mazes, averaged over 10 random instances. As B and W increase, the **mean absolute Ricci curvature**, its **standard deviation**, and **range** all grow—indicating more curved local geometry. **Entropy difference** also rises, showing increased bias in random walk exploration. The **coverage ratio** α increases as well, reflecting less efficient exploration. In short, higher (B, W) values yield more complex and less navigable environments.

4.2.2 Tabular Atari

Ricci Statistic	EH	ROP	MSAO
Min	-0.71	0.52	0.62
Max	0.54	-0.66	-0.65
Mean	-0.46	0.31	0.59
Mean($\lvert \cdot \rvert$)	0.55	-0.62	-0.63
STD	0.63	-0.41	-0.58
Range	0.68	-0.53	-0.62

Table 2: Spearman correlation coefficients between various statistics of Ricci curvature (row entries) and complexity measures (column entries) in tabular Atari, based on metrics introduced by Laidlaw et al. (2023). **EH**: Effective Horizon, **ROP**: Reward Optimality Probability, **MSAO**: Minimum State-Action Occupancy.

Table 2 reports Spearman correlations between Ricci curvature statistics and standard complexity measures in tabular Atari environments Laidlaw et al. (2023). Minimum curvature correlates negatively with EH and positively with ROP and MSAO, linking negative curvature to local divergence, randomness, and limited coverage. In contrast, maximum curvature shows the opposite trend, suggesting deeper planning but less exploratory diversity. Curvature variability (mean absolute value, standard deviation, range) also correlates positively with EH and negatively with ROP and MSAO, indicating that heterogeneous geometry demands deeper planning and fewer optimal

choices. Overall, negative curvature signals unpredictability, while positive curvature reflects local structure—highlighting Ricci curvature as a meaningful indicator of environment complexity.

4.3 RQ3: Are ORC values useful as an intrinsic reward?

As shown earlier, under a random policy, regions with negative ORC (e.g., bridges and branching points) promote exploration (**should be visited more**), while regions with high positive ORC hinder it due to excessive local connectivity (**should be visited less**). This motivates using $-\kappa^{\pi_u}(s)$—the ORC under a uniform random policy π_u—as an intrinsic reward: states with highly negative ORC yield large positive rewards (**encouraging** the agent to visit them more), and those with highly positive ORC yield large negative rewards (**discouraging** the agent to visit them more). In this subsection, we evaluate an agent trained with $-\kappa^{\pi_u}(s)$ as its intrinsic reward, analyze its exploration behavior, and compare the Ricci curvature of its induced policy to that of the random policy. We also compare this with a count-based reward, $\frac{1}{\text{State Count}}$, which encourages visiting rarely explored states. To compare these policies, we have used the following evaluation metrics:

Coverage Uniformity: (i) *Entropy of normalized state visitations*: Measures how evenly the agent explores the state space (higher is better). (ii) Δ*Entropy*: Difference between uniform entropy and observed entropy (lower is better).

Coverage Speed: (iii) α *(Normalized time to 90% coverage)*: Steps to reach 90% of states, normalized by total state count (lower is better).

To ensure a comprehensive comparison, we use high-complexity mazes ($B = 0.9$, $W = 0.9$) of three different sizes and Tabular Atari games (Laidlaw et al., 2023). Experimental details are in Appendix D. Unlike Sections 4.1–4.2, which merged all directions at each location for visualization, here we use the full state space (agent position and orientation). Corresponding non-merged Ricci curvature maps are shown in Figures 16–18 in Appendix G.

Table 3 shows that using $-$Ricci as an intrinsic reward significantly lowers average ORC values compared to both count-based and random policies. This aligns with expectations, as the agent avoids highly connected regions (with large positive curvature), making them less connected under the learned policy. Notably, the ORC range increases while the standard deviation decreases or remains stable—implying values are more concentrated around zero. Although extremes become more pronounced, the mean absolute curvature drops, suggesting that highly curved regions flatten. This is desirable as the overall geometry of the environment becomes flatter (as exploration in flat regions is more efficient and uniform) under the learned policy. Visualizations are provided in Appendix G, Figures 16–18.

Room Size	Policy	Mean ($\rightarrow 0$)	Mean($\lvert ORC \rvert$) $\downarrow$	STD $\downarrow$	Range $\downarrow$
15x15		0.09 ± 0.01	0.16 ± 0.00	0.19 ± 0.01	$\mathbf{0.93} \pm 0.10$
		0.23 ± 0.03	0.26 ± 0.02	0.26 ± 0.01	1.39 ± 0.07
		$\mathbf{0.04} \pm 0.01$	$\mathbf{0.14} \pm 0.00$	$\mathbf{0.17} \pm 0.01$	1.20 ± 0.05
21x21		0.14 ± 0.01	0.20 ± 0.01	0.20 ± 0.00	$\mathbf{0.95} \pm 0.05$
		0.36 ± 0.04	0.37 ± 0.03	0.26 ± 0.01	1.41 ± 0.12
		$\mathbf{0.06} \pm 0.01$	$\mathbf{0.16} \pm 0.01$	$\mathbf{0.19} \pm 0.00$	1.36 ± 0.16
31x31		0.22 ± 0.00	0.26 ± 0.00	$\mathbf{0.20} \pm 0.00$	$\mathbf{0.99} \pm 0.02$
		0.61 ± 0.01	0.61 ± 0.00	0.13 ± 0.01	1.14 ± 0.09
		$\mathbf{0.05} \pm 0.01$	$\mathbf{0.16} \pm 0.00$	$\mathbf{0.20} \pm 0.00$	1.69 ± 0.06

Random Policy $IR = -$Ricci $IR = \frac{1}{\text{State Count}}$

$\downarrow$: Lower better, $\rightarrow 0$: Closer to zero is better

Mean($\lvert ORC \rvert$): Average of the absolute values of ORC across all states. This ensures that the reduction in **Mean** is not simply due to an increase in large negative ORC values.

Table 3: Statistics of ORC calculated by random walks performed under different policies.

Table 4 shows that the agent trained with $-$Ricci as an intrinsic reward consistently outperforms both the random policy and the agent trained with count-based intrinsic reward across nearly all metrics

Room Size	Policy	Entropy of Normalized State Visitations ↑	ΔEntropy ↓	α ↓
15x15		8.25 ± 0.04	0.35 ± 0.04	60.14 ± 27.17
		8.25 ± 0.12	0.35 ± 0.11	79.29 ± 26.10
		8.35 ± 0.04	**0.25** ± 0.03	**47.25** ± 30.10
21x21		9.19 ± 0.01	0.45 ± 0.01	68.72 ± 7.95
		9.27 ± 0.05	0.37 ± 0.05	51.83 ± 11.34
		9.35 ± 0.02	**0.29** ± 0.02	**48.23** ± 5.79
31x31		9.98 ± 0.17	0.78 ± 0.20	NA
		9.95 ± 0.25	0.83 ± 0.25	NA
		10.05 ± 0.22	**0.73** ± 0.22	**49.77** ± 10.20
Tabular Atari (Laidlaw et al., 2023)		6.16 ± 1.10	2.90 ± 1.31	102.35 ± 46.73
		6.17 ± 1.02	**2.75** ± 1.26	110.72 ± 49.89
		6.12 ± 1.11	2.80 ± 1.33	**94.76** ± 44.54

Random Policy $IR = -\text{Ricci}$ $IR = \frac{1}{\text{State Count}}$

↑: Higher better, ↓: Lower better, **NA**: 90% coverage not reached

Δ Entropy $= \log_2(N_{\text{States}}) - H(\text{Normalized State Visitations})$

Table 4: Entropy and coverage statistics across environments and intrinsic reward strategies. See legend above for IR type.

and environments. Notably, in the 31×31 maze, neither the random nor the count-based agent was able to cover 90% of the states within 100k steps. This highlights the strength of our method in large, complex environments where efficient exploration is critical. While our method also performs better on average in the Tabular Atari environments, the margin is smaller—particularly for the entropy metric, where the count-based method slightly outperforms ours. This can be attributed to the frequent environment restarts in the Tabular Atari dataset, which limit the length of Markov chains and reduce the potential benefit of curvature-guided exploration. In contrast, mazes involve no restarts, so agents that get trapped in highly connected regions (as with a random policy) remain there for longer, amplifying the advantage of our method which prevents being trapped in highly connected regions. An extended performance comparison is provided in Appendix G.

5 Related Work

Existing RL complexity measures have key limitations: 1) They often yield only global scores (e.g., Effective Horizon (Laidlaw et al., 2023)); 2) Many depend on specific algorithms (Aitchison et al., 2023), entangling complexity with rewards and agent performance; and 3) Several require costly pipelines like duplication pruning or post-training analysis (Laidlaw et al., 2023; Aitchison et al., 2023). In contrast, our approach: 1) Offers a local, geometry-aware measure via Ricci curvature; 2) Is reward-agnostic; and 3) Links Ollivier-Ricci curvature (ORC) to the successor representation (SR), making it compatible with any RL method using or approximating SR. Even without SR, ORC can be estimated directly from the state-transition graph built from the agent's experience. The use of Ricci curvature in RL is still limited. Existing work is either theoretical (Nedergaard & Morales, 2025) or tailored to specific domains such as navigation (Song & Lee, 2024). Our work introduces a general, practical approach for estimating and applying ORC in reward-free environments, demonstrating its utility for exploration. This work also relates to intrinsic reward, first proposed by Schmidhuber (1991) and later extended through entropy-maximizing methods that encourage uniform state visitation via state-counting (Burda et al., 2018; Liu & Abbeel, 2021; Bellemare et al., 2016; Hazan et al., 2019). While effective, these approaches often ignore the environment's geometric structure. In contrast, our curvature-based reward captures long-range structural information, offering a richer exploratory signal. Importantly, our method is orthogonal to existing intrinsic rewards and can be integrated with them seamlessly.

6 Conclusion and Future Work

In this paper, we introduced Ollivier-Ricci Curvature (ORC) as a metric to capture local complexity in reinforcement learning environments and demonstrated that its statistics also reflect global structural complexity. We showed that using ORC as an intrinsic reward in reward-free settings substantially improves exploration compared to random walk and count-based methods, particularly in complex environments with trapping states. ORC's strength lies in integrating successor representation with local connectivity graphs, providing complementary global and local perspectives. Future work will extend this framework to continuous environments using successor features and online computation, and investigate how ORC interacts with other intrinsic and extrinsic rewards to develop more effective exploration strategies.

References

Matthew Aitchison, Penny Sweetser, and Marcus Hutter. Atari-5: Distilling the arcade learning environment down to five games. In *International Conference on Machine Learning*, pp. 421–438. PMLR, 2023.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

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

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

Elad Hazan, Sham Kakade, Karan Singh, and Abby Van Soest. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, pp. 2681–2691. PMLR, 2019.

Cassidy Laidlaw, Stuart J Russell, and Anca Dragan. Bridging rl theory and practice with the effective horizon. *Advances in Neural Information Processing Systems*, 36:58953–59007, 2023.

Hao Liu and Pieter Abbeel. Behavior from the void: Unsupervised active pre-training. *Advances in Neural Information Processing Systems*, 34:18459–18473, 2021.

Alexander Nedergaard and Pablo A Morales. An information-geometric approach to artificial curiosity. *arXiv preprint arXiv:2504.06355*, 2025.

Chien-Chun Ni, Yu-Yao Lin, Feng Luo, and Jie Gao. Community detection on networks with ricci flow. *Scientific reports*, 9(1):1–12, 2019.

Yann Ollivier. Ricci curvature of markov chains on metric spaces. *Journal of Functional Analysis*, 256(3):810–864, 2009.

MMG Ricci and Tullio Levi-Civita. Méthodes de calcul différentiel absolu et leurs applications. *Mathematische Annalen*, 54(1):125–201, 1900.

Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pp. 222–227, 1991.

Wongeun Song and Jungwoo Lee. Ricci planner: Zero-shot transfer for goal-conditioned reinforcement learning via geometric flow. *IEEE Access*, 12:24027–24038, 2024.

Cédric Villani et al. *Optimal transport: old and new*, volume 338. Springer, 2008.

Appendix

A Abbreviations

Abbreviation	Expanded
ORC	Ollivier-Ricci Curvature
SR	Successor Representation
IR	Intrinsic Reward
EH	Effective Horizon
ROP	Reward Optimality Probability
MSAO	Minimum State-Action Occupancy

Table 5: Table of abbreviations used in this paper.

B Offline Estimation of SR and ORC

B.1 Pseudoscode for SR Offline Calculation

Algorithm 1 Estimate Successor Representation $\mathbf{SR}^\pi$

Require: Set of states $\mathcal{S}$, discount factor $\gamma \in (0, 1]$, walk length L_{SR}, number of walks N_{SR}, policy $\pi(a \mid s)$

Ensure: Successor representation matrix $\mathbf{SR}^\pi \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$

```
Initialize SR^π ← 0^{|S|×|S|}
for each s ∈ S do
    for i = 1 to N_SR do
        s_curr ← s
        for t = 0 to L_SR − 1 do
            SR^π[s, s_curr] ← SR^π[s, s_curr] + γ^t
            Sample action a ~ π(· | s_curr)
            s_curr ← Take Action(a)
        end for
    end for
    SR^π[s, :] ← SR^π[s, :]/N_SR
end for
return SR^π
```

B.2 Pseudoscode for ORC Offline Calculation

Algorithm 2 Estimate Ollivier-Ricci Curvature κ^π

Require: Set of states $\mathcal{S}$, policy $\pi(a \mid s)$, number of walks N_{ORC}, walk length L_{ORC}
Ensure: Curvature values $\kappa^\pi(s)$ for all $s \in \mathcal{S}$

```
Initialize adjacency matrix A ← ∞^{|S|×|S|}                ▷ Initialize with no edges
for each s ∈ S do
    for i = 1 to N_ORC do
        s_curr ← s
        for t = 1 to L_ORC do
            Sample action a ~ π(· | s_curr)
            s_next ← transition(s_curr, a)
            if A[s, s_next] > t then
                A[s, s_next] ← t
            end if
            s_curr ← s_next
        end for
    end for
end for
                                                            ▷ Now compute ORC using Equation 1
for each s ∈ S do
    Let N(s) ← {s' | A[s, s'] < ∞}
    for each s' ∈ N(s) do
        Compute κ^π(s, s') using Equation 1
    end for
    κ^π(s) ← (1/|N(s)|) Σ_{s'∈N(s)} κ^π(s, s')
end for
return κ^π(s) for all s
```

C Online Estimation of SR and ORC

In this section, we describe how to estimate the successor representation (SR) and Ollivier-Ricci Curvature (ORC) online, during the agent's interaction with the environment. Unlike the offline method, which requires iterating over all states and resetting the environment multiple times, the online approach updates SR and ORC incrementally as the agent explores.

C.1 Online SR Calculation

Instead of starting random walks from every state beforehand, we update the SR matrix progressively during the agent's trajectory. At each time step t, the agent observes the current state s_t and updates the SR row corresponding to the states visited in the recent past s_{t-k} for $k = 0, \ldots, L_{SR} - 1$, discounting by γ^k.

Algorithm 3 Online Estimation of Successor Representation $\mathbf{SR}^\pi$

Require: Discount factor $\gamma \in (0, 1]$, max trace length L_{SR}, environment $\mathcal{E}$, policy π
Ensure: Successor representation matrix $\mathbf{SR}^\pi \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$

```
Initialize SR^π ← 0^{|S|×|S|}
Initialize an empty FIFO queue Q to store recent states (max length L_SR)
for each episode do
    Reset environment, observe initial state s_0
    Clear queue Q, enqueue s_0
    for each time step t do
        Sample action a_t ~ π(· | s_t)
        Execute a_t, observe next state s_{t+1}
        Enqueue s_{t+1} into Q (drop oldest if full)
        for k = 0 to min(t, L_SR − 1) do
            s_start ← Q[index |Q| − 1 − k]                ▷ state visited k steps ago
            SR^π[s_start, s_{t+1}] ← SR^π[s_start, s_{t+1}] + γ^k
        end for
        s_t ← s_{t+1}
    end for
end for
return SR^π
```

C.2 Online ORC Calculation

After estimating the successor representation (SR) online, we incrementally build a connectivity graph over the state space to estimate Ollivier-Ricci Curvature (ORC) during agent interaction. This graph approximates geodesic distances needed for computing Wasserstein distances in Equation 1 using shortest-path lengths.

At each time step, we update the adjacency matrix based on observed transitions. Specifically, when the agent transitions from state s to s' at step t in the current episode, we record the shortest known path length between these states. If the current recorded distance is greater than t, it is updated to t.

Once enough transitions have been observed, ORC $\kappa^\pi(s, s')$ is computed between connected states using the current adjacency matrix and Equation 1. The local curvature $\kappa^\pi(s)$ at each state is the average curvature over all neighbors.

Algorithm 4 Online Estimation of Ollivier-Ricci Curvature κ^π

Require: Set of states $\mathcal{S}$, policy $\pi(a \mid s)$, max episode length L_{ORC}
Ensure: Curvature values $\kappa^\pi(s)$ for all $s \in \mathcal{S}$

```
Initialize adjacency matrix A ← ∞^{|S|×|S|}                    ▷ No edges initially
for each episode do
    Reset environment, observe initial state s_0
    s_curr ← s_0
    for each time step t = 1, ..., L_ORC do
        Sample action a_t ~ π(· | s_curr)
        Execute a_t, observe next state s_next
        if A[s_curr, s_next] > t then
            A[s_curr, s_next] ← t
        end if
        s_curr ← s_next
    end for
end for
                                    ▷ Compute Ollivier-Ricci curvature using Equation 1
for each s ∈ S do
    Let N(s) ← {s' | A[s, s'] < ∞}
    for each s' ∈ N(s) do
        Compute κ^π(s, s') using Equation 1
    end for
    κ^π(s) ← (1/|N(s)|) Σ_{s'∈N(s)} κ^π(s, s')
end for
return κ^π(s) for all s
```

D Hyperparameters

Table 6 shows the parameters used in different parts of the paper.

Table 6: Hyperparameters for Maze Environment and Q-Learning Experiments

Parameter	Value	Description
Maze Environment		
Size	[15, 21, 31]	Room Size
W	[0.1, 0.4, 0.9]	Winding Factor
B	[0.1, 0.4, 0.9]	Branching Factor
Seed	[42, 13, 4242, 1313]	Seed for reproducibility in maze generation
SR Calculation		
N_{SR}	1000	Number of experiments (episodes)
γ_{SR}	0.99	Discount factor for SR calculation
L_{SR}	30	Horizon for SR calculation
ORC Calculation		
N_{ORC}	1000	Number of experiments (episodes)
L_{ORC}	5	Horizon for connectivity graph construction
α_{ORC}	0.0	Idleness Parameter
τ	[0.1 (Atari), 10.0(Mazes)]	Temperature to obtain stochastic policy from Q-values.
Q-Learning Parameters		
α	0.1	Learning rate
γ	0.99	Discount factor
ϵ	0.1	Exploration rate
N	5000	Number of Q-Learning Episodes
M	100	Number of steps per episode
Exploration Evaluation		
N_{exp}	[1000(Atari), 10(Mazes)]	Number of episodes
M_{exp}	[1000(Atari), 10000(Mazes)]	Number of steps in each episode

E Bigger Mazes

The plots corresponding to larger mazes are presented in Figure 5. As evident from the figure, increasing the maze size leads to a higher number of branching points and dead-ends. This added complexity makes it harder for the agent to navigate, which can affect how it explores and learns in the environment.

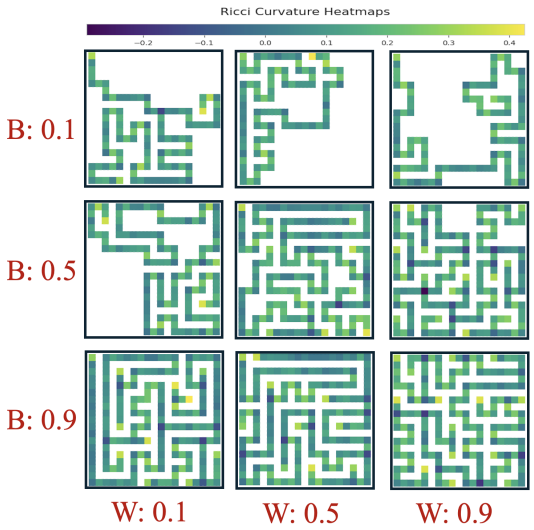

Figure 5: Ricci curvature values across different locations in mazes (21x21) with varying **B**ranching and **W**inding factors. As observed, dead-ends (end of branches) and winding segments tend to exhibit large positive Ricci values, branching points correspond to large negative values, and straight corridors typically have Ricci values close to zero.

F Tabular Atari Ricci Values

In this section, we see examples from various Atari games (Figure 6-15). These frames are sorted based on the Ricci curvature of the middle frame. On the figures, the ricci value of the middle frame is written with red font on top of the middle frame. These figures show how at states with negative ORC trajectories start diverging and how at states with positive ORC trajectories start converging.

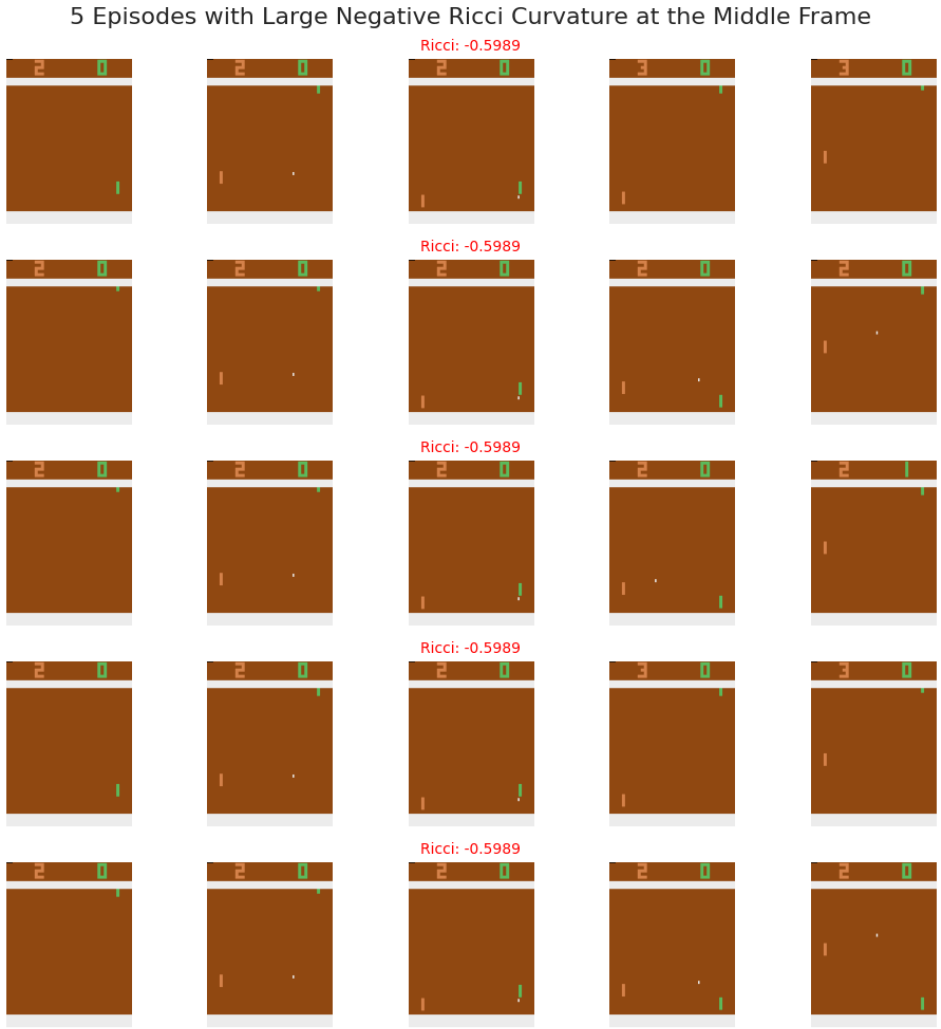

Figure 6: Examples of frame sequences from different trials of game "Pong" where the middle frame has a large negative Ricci curvature. While the middle frames appear visually similar, the subsequent frames diverge significantly, resembling diverging random walks.

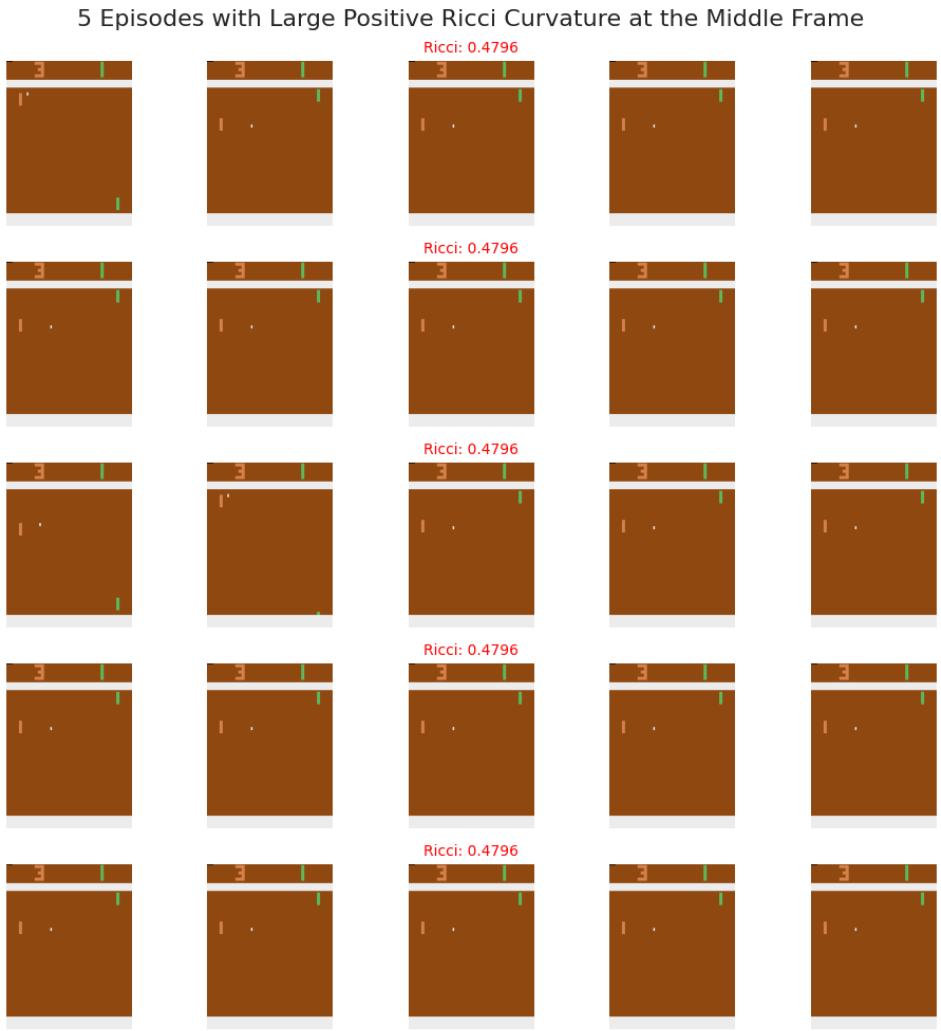

Figure 7: Examples of frame sequences from different trials of game "Pong" where the middle frame has a large positive Ricci curvature. Both the middle and subsequent frames exhibit high similarity across trials, reflecting converging behavior.

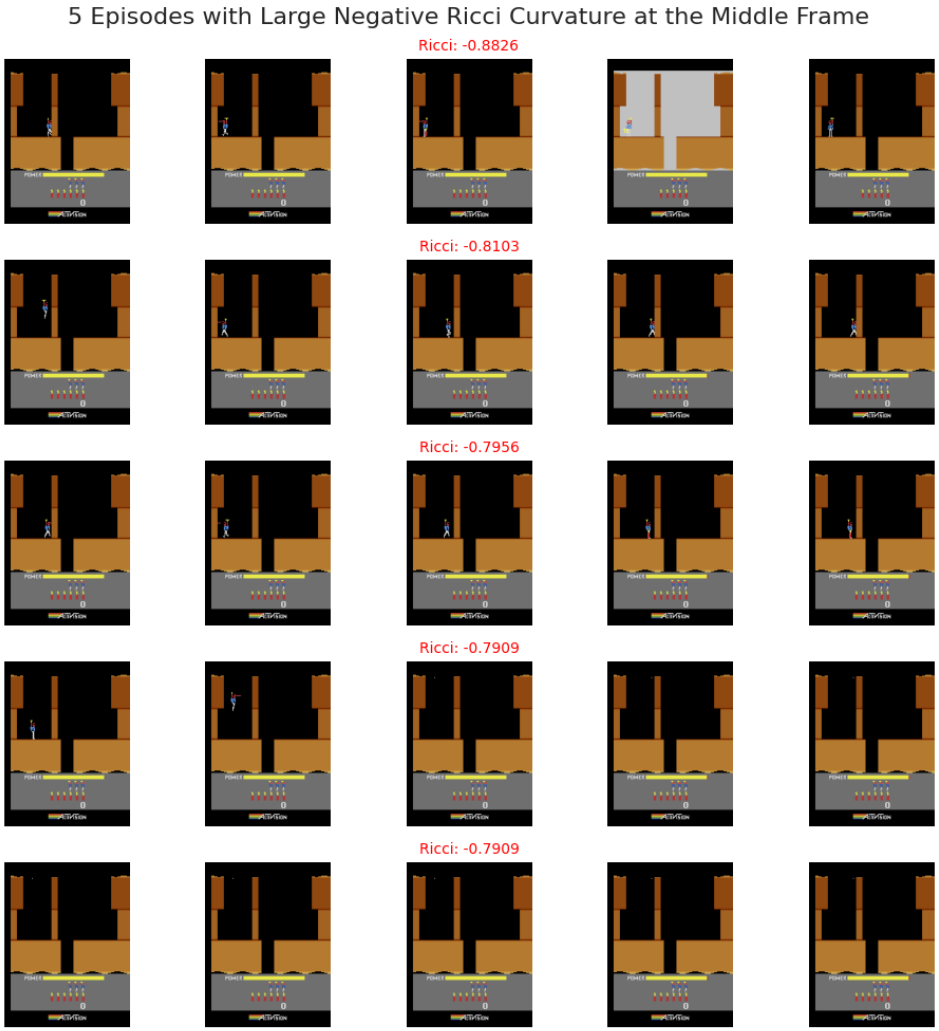

Figure 8: Examples of frame sequences from different trials of game "Hero" where the middle frame has a large negative Ricci curvature. While the middle frames appear visually similar, the subsequent frames diverge significantly, resembling diverging random walks.

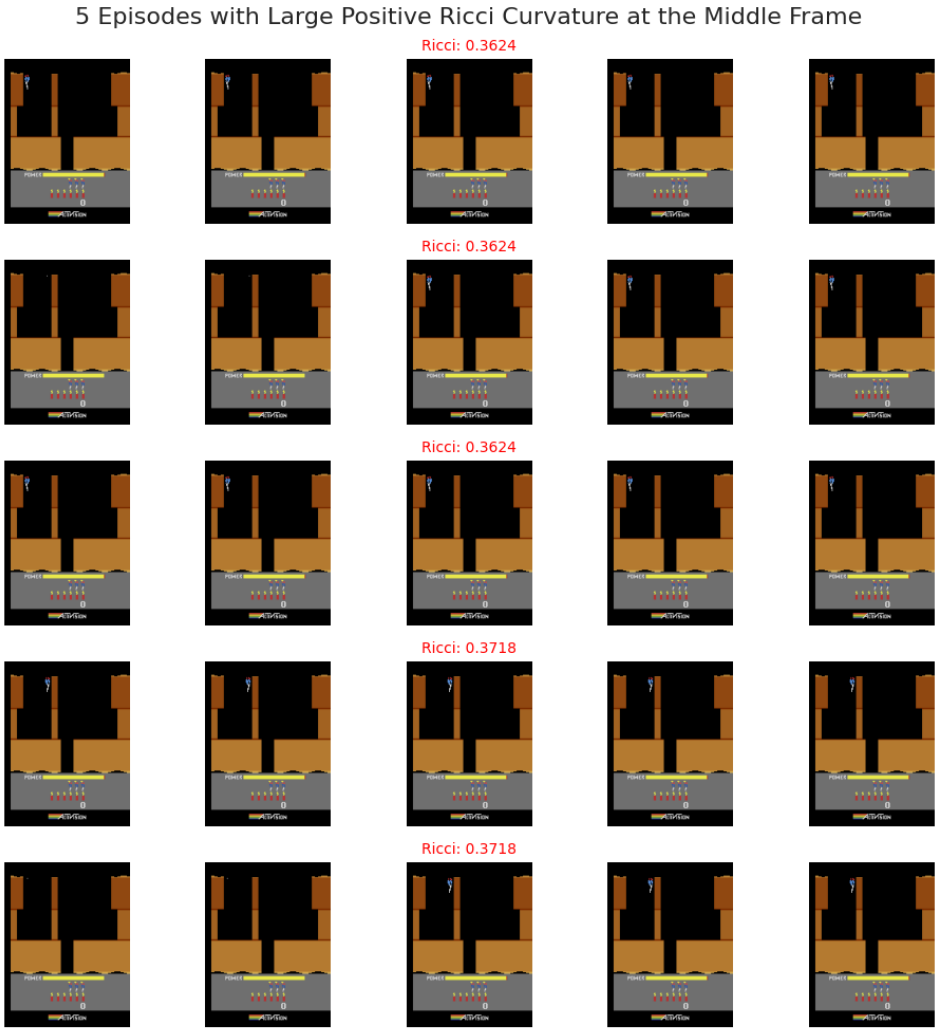

Figure 9: Examples of frame sequences from different trials of game "Hero" where the middle frame has a large positive Ricci curvature. Both the middle and subsequent frames exhibit high similarity across trials, reflecting converging behavior.

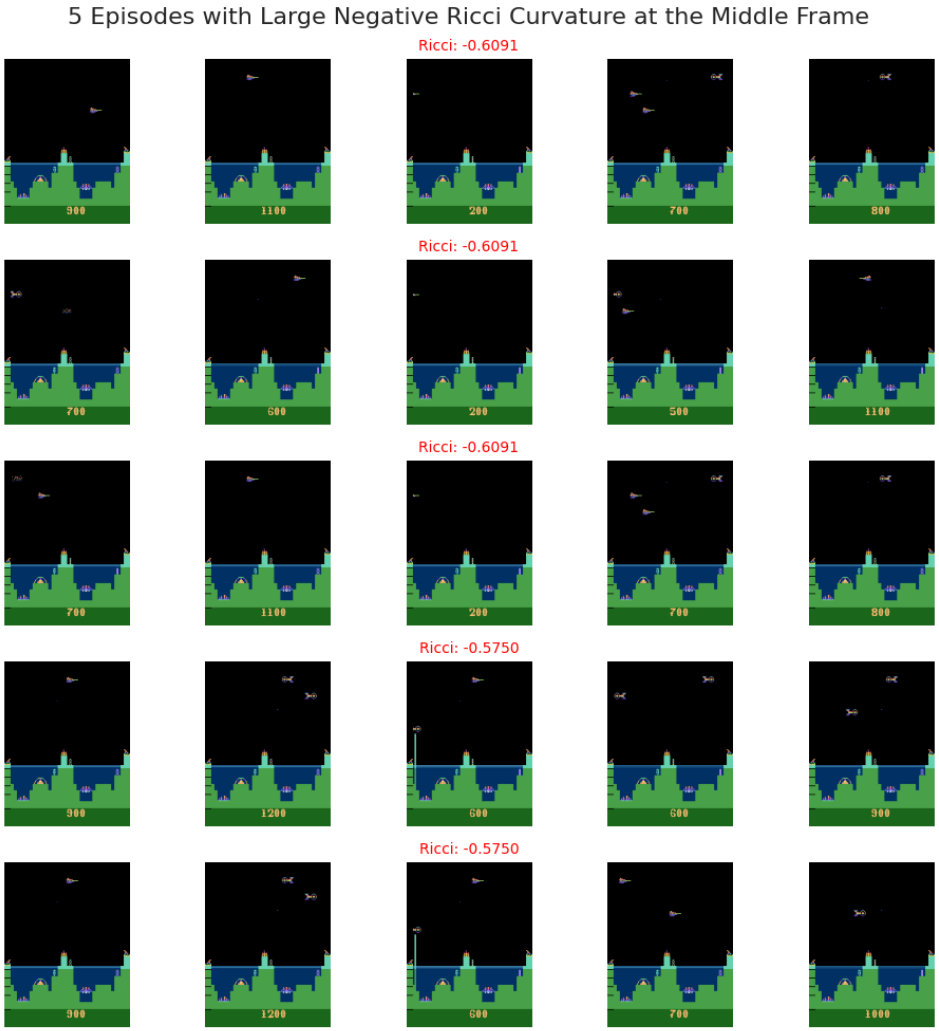

Figure 10: Examples of frame sequences from different trials of game "Atlantis" where the middle frame has a large negative Ricci curvature. While the middle frames appear visually similar, the subsequent frames diverge significantly, resembling diverging random walks.

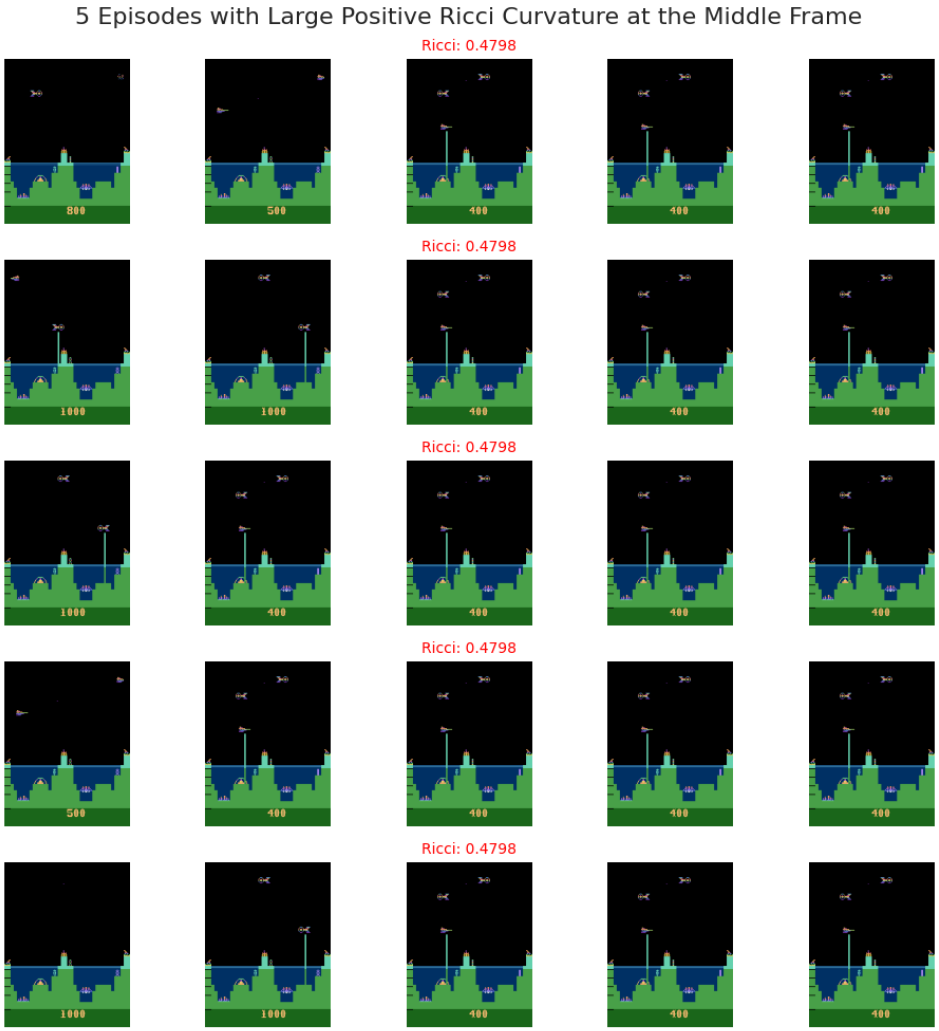

Figure 11: Examples of frame sequences from different trials of game "Atlantis" where the middle frame has a large positive Ricci curvature. Both the middle and subsequent frames exhibit high similarity across trials, reflecting converging behavior.

Figure 12: Examples of frame sequences from different trials of game "Breakout" where the middle frame has a large negative Ricci curvature. While the middle frames appear visually similar, the subsequent frames diverge significantly, resembling diverging random walks.

Figure 13: Examples of frame sequences from different trials of game "Breakout" where the middle frame has a large positive Ricci curvature. Both the middle and subsequent frames exhibit high similarity across trials, reflecting converging behavior.

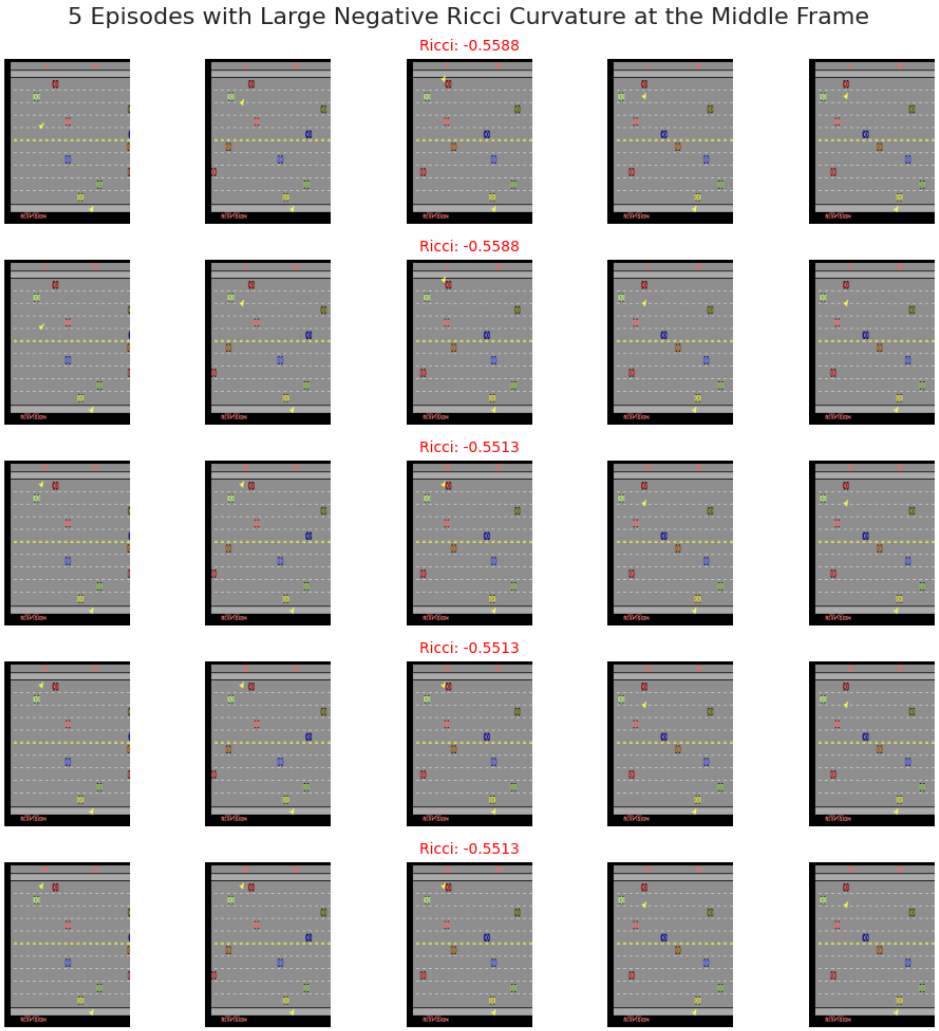

Figure 14: Examples of frame sequences from different trials of game "Freeway" where the middle frame has a large negative Ricci curvature. While the middle frames appear visually similar, the subsequent frames diverge significantly, resembling diverging random walks.

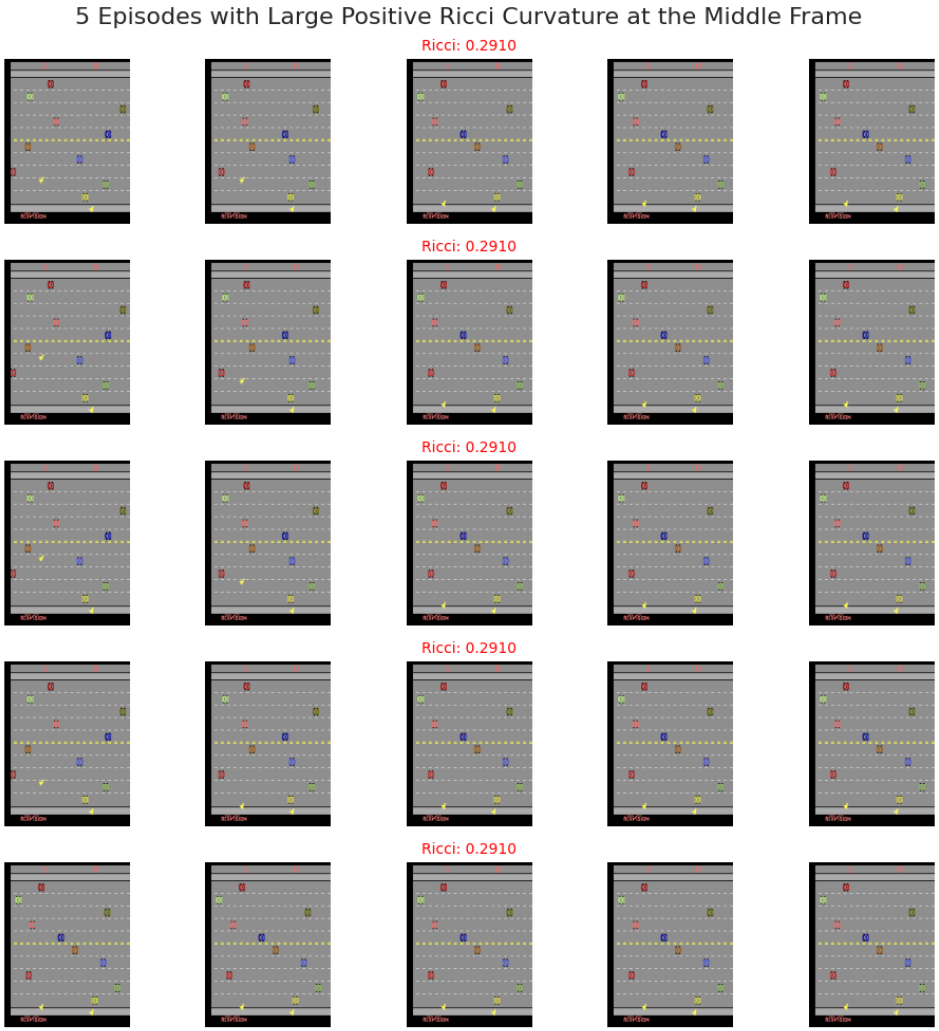

Figure 15: Examples of frame sequences from different trials of game "Freeway" where the middle frame has a large positive Ricci curvature. Both the middle and subsequent frames exhibit high similarity across trials, reflecting converging behavior.

G Extended Performance Comparison

In this appendix, we present extended experimental results related to Section 4.3, as shown in Tables 7 and 8. We also include visualizations illustrating how Ollivier-Ricci Curvature (ORC) values change under policies trained with the two intrinsic reward methods discussed compared to random policy. Additionally, we show how state coverage evolves over time under different policies.

Table 7 presents additional statistics on the Ollivier-Ricci Curvature (ORC) values computed under different policies. As discussed in Section 4.3, using the negative of ORC as an intrinsic reward leads to a significant reduction in the average of Ricci values, indicating that the resulting policy induces a flatter environment. This reduction is not merely due to the inclusion of more negative values, as the average absolute Ricci value also decreases. Furthermore, the standard deviation is reduced or unchanged, suggesting that the values are more concentrated around zero. While the range and maximum values increase and the minimum becomes more negative—likely due to a few outlier states—our focus is on the average and standard deviation of Ricci values, which better reflect the overall flattening of the environment under the trained policy.

Room Size	Policy	Min ($\rightarrow 0$)	Max ($\rightarrow 0$)	Mean ($\rightarrow 0$)	Mean($\lvert ORC \rvert$) $\downarrow$	STD $\downarrow$	Range $\downarrow$
15x15		-0.34 $\pm$ 0.06	0.58 $\pm$ 0.06	0.09 $\pm$ 0.01	0.16 $\pm$ 0.00	0.19 $\pm$ 0.01	0.93 $\pm$ 0.10
		-0.49 $\pm$ 0.07	0.90 $\pm$ 0.01	0.23 $\pm$ 0.03	0.26 $\pm$ 0.02	0.26 $\pm$ 0.01	1.39 $\pm$ 0.07
		-0.65 $\pm$ 0.06	0.55 $\pm$ 0.06	0.04 $\pm$ 0.01	0.14 $\pm$ 0.00	0.17 $\pm$ 0.01	1.20 $\pm$ 0.05
21x21		-0.32 $\pm$ 0.02	0.61 $\pm$ 0.03	0.14 $\pm$ 0.01	0.20 $\pm$ 0.01	0.20 $\pm$ 0.00	0.95 $\pm$ 0.05
		-0.50 $\pm$ 0.12	0.91 $\pm$ 0.00	0.36 $\pm$ 0.04	0.37 $\pm$ 0.03	0.26 $\pm$ 0.01	1.41 $\pm$ 0.12
		-0.68 $\pm$ 0.08	0.67 $\pm$ 0.08	0.06 $\pm$ 0.01	0.16 $\pm$ 0.01	0.19 $\pm$ 0.00	1.36 $\pm$ 0.16
31x31		-0.32 $\pm$ 0.01	0.68 $\pm$ 0.01	0.22 $\pm$ 0.00	0.26 $\pm$ 0.00	0.20 $\pm$ 0.00	0.99 $\pm$ 0.02
		-0.23 $\pm$ 0.09	0.91 $\pm$ 0.00	0.61 $\pm$ 0.01	0.61 $\pm$ 0.00	0.13 $\pm$ 0.01	1.14 $\pm$ 0.09
		-0.78 $\pm$ 0.05	0.91 $\pm$ 0.00	0.05 $\pm$ 0.01	0.16 $\pm$ 0.00	0.20 $\pm$ 0.00	1.69 $\pm$ 0.06

Random Policy $IR = -\text{Ricci}$ $IR = \frac{1}{\text{State Count}}$

$\downarrow$: Lower better, $\rightarrow 0$: Closer to zero is better

Mean($\lvert ORC \rvert$): Average of the absolute values of ORC across all states. This ensures that the reduction in **Mean** is not simply due to an increase in large negative ORC values.

Table 7: Statistics of ORC calculated by random walks based on different policies.

Table 8 provides an extended version of Table 4, including experiments conducted on a 21×21 maze with loops that introduce small room-like structures. As observed, exploration in this environment is easier compared to the corresponding maze of the same size without loops. This improvement is reflected in higher entropy, a smaller gap between the achieved entropy and that of a uniform distribution, and a faster time to reach 90% state coverage.

Room Size	Policy	Entropy of Normalized State Visitations ↑	ΔEntropy ↓	α ↓	Steps for 90% Coverage ↓
15x15		8.25 ± 0.04	0.35 ± 0.04	60.14 ± 27.17	23,334 ± 10,542
		8.25 ± 0.12	0.35 ± 0.11	79.29 ± 26.10	30,764 ± 10,107
		8.35 ± 0.04	**0.25** ± 0.03	**47.25** ± 30.10	**18,333** ± 11,569
21x21		9.19 ± 0.01	0.45 ± 0.01	68.72 ± 7.95	54,701 ± 6,318
		9.27 ± 0.05	0.37 ± 0.05	51.83 ± 11.34	41,256 ± 9,123
		9.35 ± 0.02	**0.29** ± 0.02	**48.23** ± 5.79	**38,391** ± 4,607
21x21 + Loops		9.36 ± 0.08	0.35 ± 0.07	48.80 ± 19.40	40,796 ± 16,218
		9.42 ± 0.14	0.29 ± 0.13	37.24 ± 16.39	31,132 ± 13,702
		9.49 ± 0.12	**0.22** ± 0.12	**23.33** ± 5.09	**19,503** ± 4,255
31x31		9.98 ± 0.17	0.78 ± 0.20	NA	NA
		9.95 ± 0.25	0.83 ± 0.25	NA	NA
		10.05 ± 0.22	**0.73** ± 0.22	**49.77** ± 10.20	**87,755** ± 18,109
Tabular Atari (Laidlaw et al., 2023)		6.16 ± 1.10	2.90 ± 1.31	102.35 ± 46.73	452,989 ± 254,366
		6.17 ± 1.02	**2.75** ± 1.26	110.72 ± 49.89	484,578 ± 270,583
		6.12 ± 1.11	2.80 ± 1.33	**94.76** ± 44.54	**425,925** ± 255,187

Random Policy $IR = -\text{Ricci}$ $IR = \frac{1}{\text{State Count}}$

↑: Higher better, ↓: Lower better, **NA**: 90% coverage not reached (for all or some of the seed values)

Δ Entropy $= \log_2(N_{\text{States}}) - H(\text{Normalized State Visitations})$

Table 8: Entropy and coverage statistics across environments and intrinsic reward strategies. See legend above for IR type.

Figures 16, 17, and 18 present visualizations of Ollivier-Ricci Curvature (ORC) values and state coverage under different policies. In the left column of each figure, we show the ORC values across locations and directions (each state is defined by both position and orientation). White arrows indicate states with negative curvature, while red arrows show positive curvature. Dark blue corresponds to large negative values and yellow to large positive values. Similar to what we observed in Section 4.1, dead ends tend to have large positive Ricci values, while branching points exhibit large negative values.

The middle column displays ORC values induced by policies trained with $-Ricci$ (top) and $\frac{1}{\text{State Count}}$ (bottom) as intrinsic rewards. The negative ORC-based policy results in a more uniform and on average close to zero distribution of ORC values, whereas the $\frac{1}{\text{State Count}}$ policy produces areas of high visitation and unexplored regions (white squares).

The right column compares the progression of unique state coverage over time: the blue line represents the trained policy, and the orange line corresponds to a random policy. Faster coverage indicates more efficient exploration.

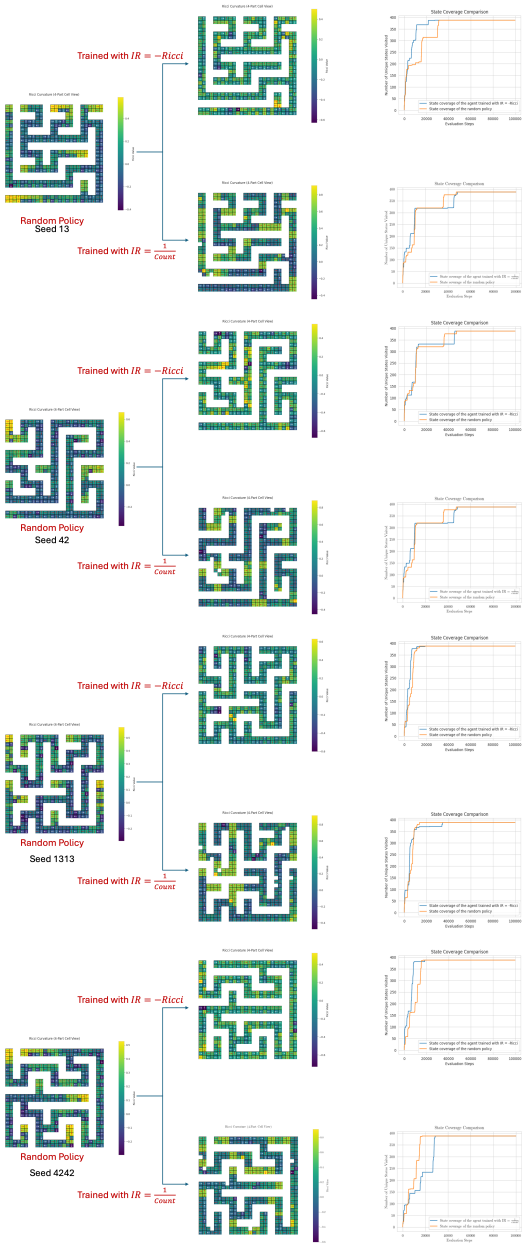

Figure 16: 15x15

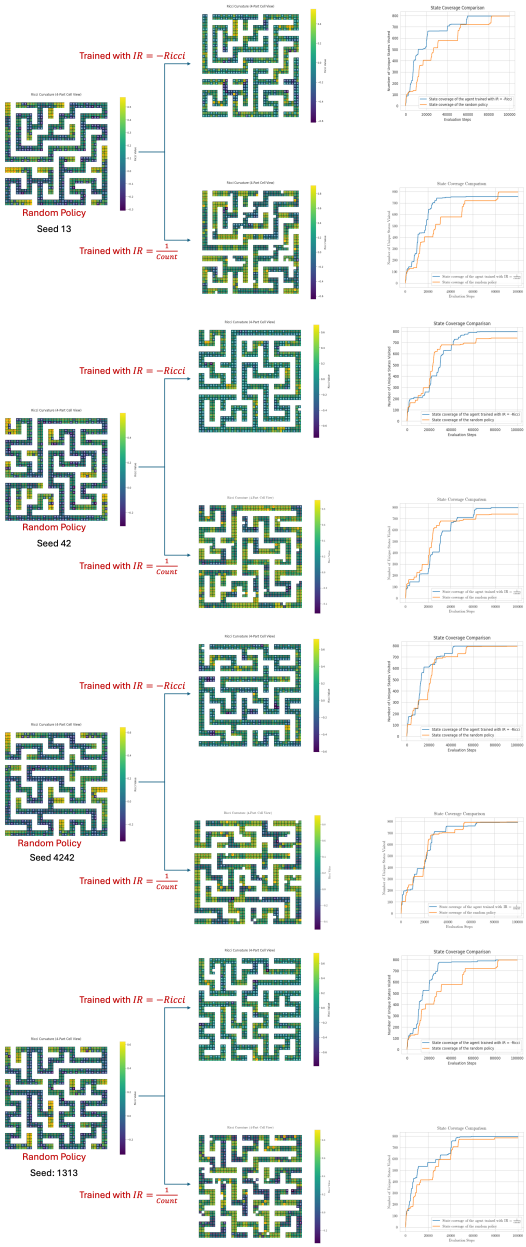

Figure 17: 21x21

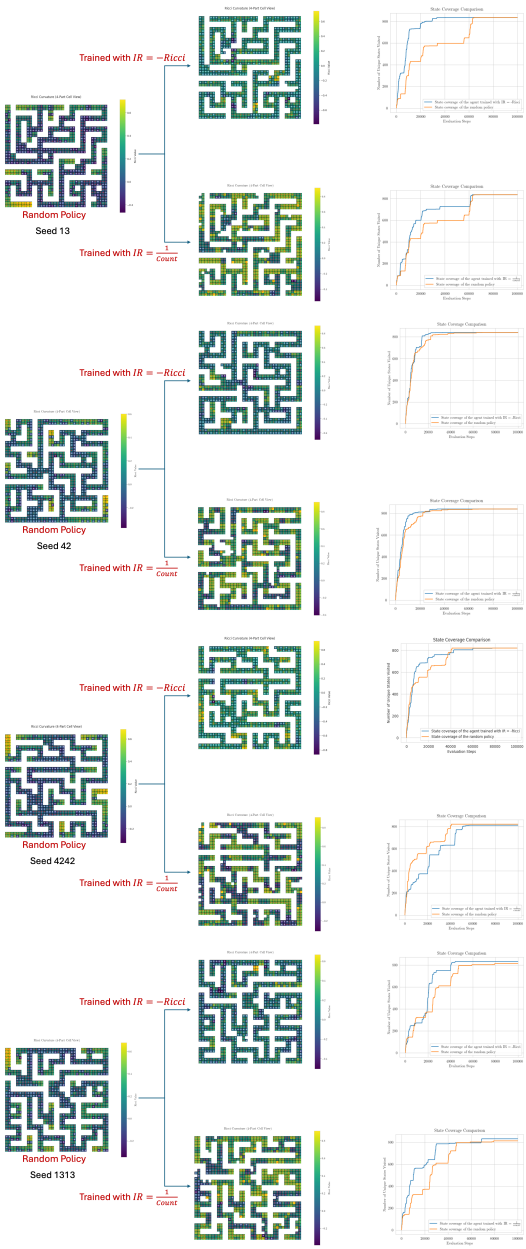

Figure 18: 21x21 with loops