A Multi-Agent World Model with Self-Driven Curiosity Coordinated Exploration

Project: Deep Reinforcement Learning

Annika Treutle AT , Thomas Kling TK , Luca Sailer LS

July 4, 2025

Abstract

In this paper, we extend intrinsically motivated, self-aware agents to multi-agent scenarios. Building on Ha & Schmidhuber's world-model framework and Haber et al.'s self-model concept, we implement in PyBullet a shared world-model encoder (VAE) and predictor (MDN-RNN) that jointly transforms both agents' observations into latent states and forecasts their temporal dynamics. Each agent maintains its own self-model, which uses these latent representations to estimate the expected prediction error of candidate actions and thereby generates an intrinsic reward. We evaluate the impact of three exploration policies ϵ-greedy, Boltzmann, and UCB-on interaction frequency and learning progress. In the multi-agent setting, both curiosity signals and model errors rise significantly compared to single-agent experiments, underscoring the increased complexity and the promise of coordinated learning. These results offer important guidance for developing efficient exploration policies in dynamic environments.

TK LS AT

---

1 Introduction

TK

Reinforcement Learning (RL) agents have demonstrated impressive capabilities across various complex tasks. However, traditional RL approaches often face significant limitations, particularly when external rewards are sparse or non-existent. Intrinsically motivated agents, which leverage internal reward signals such as curiosity, offer a promising alternative by enabling effective exploration and learning even in the absence of clear external incentives [1, 24].

In this work, we build upon the intrinsically motivated approach introduced by Haber et al. [10], which integrates predictive world models and curiosity-driven exploration, to examine its applicability and effectiveness within multi-agent reinforcement learning scenarios. Their method utilizes two complementary neural network models: a world-model that predicts outcomes of the agents' actions and a self-model that anticipates the world-model's prediction errors. The self-model encourages agents to choose actions likely to challenge the predictive capabilities of the world-model, thereby continuously stimulating novel interactions and promoting exploratory behaviors.

To extend this concept to a multi-agent context, we implement a shared world model leveraging Variational Autoencoders (VAEs), as proposed by Ha & Schmidhuber [9], alongside recurrent neural networks (RNNs) to capture complex environmental dynamics. This shared world model is paired with individual self-models for each agent, which independently estimate intrinsic rewards, driving the exploratory actions of each agent while facilitating coordinated exploration.

Specifically, we conduct experiments within the Pybullet simulation environment, systematically investigating the impact of deploying multiple interacting agents. Our experimental setup includes variations in object shapes and environmental configurations to assess their influence on exploratory behavior and overall learning performance. We further provide a comparative analysis between single-agent and multi-agent scenarios, exploring convergence rates of the world model and the frequency and dynamics of agent interactions both with the environment and with each other.

2 Related Work

2.1 World Models

LS

The concept of agents learning internal models of their environment, often referred to as "world models" has gained significant traction within artificial intelligence research, particularly in the field of Reinforcement Learning (RL). These models represent an agent's understanding of how the environment evolves over time, enabling prediction and

planning. This subsection reviews key related concepts and foundational work relevant to our approach, focusing on the underlying principles, specific architectures, and the role of intrinsic motivation.

World models are fundamentally rooted in the paradigm of Model-Based Reinforcement Learning (MBRL) [19, 28]. Unlike model-free methods (e.g., Q-learning, Policy Gradients) that learn policies or value functions directly from environmental interaction, MBRL involves explicitly learning a model of the environment's dynamics, typically approximating the state transition function $P(s_{t+1}|s_t, a_t)$ and often the reward function $R(s_t, a_t)$ based on collected experience [18]. The primary advantage sought with MBRL is enhanced sample efficiency; by learning a model, an agent can generate simulated or "imagined" trajectories offline, significantly reducing potentially costly, time-consuming, or unsafe interaction with the real environment [18, 13]. This learned model can be used for planning future actions, generating synthetic data, or augmenting real data streams [18, 19, 28]. However, MBRL's effectiveness hinges on model accuracy, as inaccuracies can lead to planning errors and compounding errors over long rollouts, especially in complex or stochastic settings [18]. Mitigating these errors is a key research focus.

Furthermore, real-world environments are often only partially observable (POMDPs), where agents receive observations (o_t) instead of true states (s_t) [16]. Solving POMDPs optimally requires intractable belief state tracking [16]. World models with recurrent components (RNNs) offer a scalable approach, using the RNN's hidden state (h_t) to implicitly summarize history and approximate a compressed belief state, enabling tractable planning from high-dimensional observations like images [16].

A highly influential architecture was proposed by Ha and Schmidhuber [9], separating a large world model from a small controller. Their world model included a Vision Model (V), a Variational Autoencoder (VAE) [14] compressing observations o_t into a latent vector z_t, trained via reconstruction loss; and a Memory Model (M), an MDN-RNN modeling temporal dynamics $P(z_{t+1}|a_t, z_t, h_t)$ in latent space, using its RNN state h_t for memory and an MDN head to predict a Gaussian mixture distribution, capturing uncertainty. The simple Controller (C) mapped $[z_t, h_t]$ to actions a_t and could be trained via Evolution Strategies entirely within the model's "dream" [9].

Jürgen Schmidhuber has been a long-standing proponent of formalizing intrinsic motivation, starting as early as 1990.51 His theory posits that an agent's intrinsic reward, often conceptualized as "fun" or "curiosity," is directly proportional to the learning progress of its internal world model.53 The core idea is that agents are motivated to create or discover novel, surprising patterns in the data stream generated by their interaction with the world, specifically patterns that their internal model does not yet understand but can learn to predict or compress more efficiently.[23]

Haber et al. [10] used world model prediction error, guided by a self-model, to drive intrinsically motivated learning. In our work, we integrate these ideas in a multi-agent context, using a Ha & Schmidhuber-style model where intrinsic motivation derived from the shared model guides collective exploration.

Subsequent research evolved world models further: PlaNet [11] introduced the Recurrent State Space Model (RSSM) with deterministic and stochastic latent parts, enabling online latent planning (CEM) via latent overshooting. The Dreamer series (V1-V3) [12, 11, 13] adopted the RSSM but learned actor-critic policies via backpropagation through imagined latent trajectories, achieving strong performance and generality (V3) [13]. While these represent advancements, we utilize the Ha & Schmidhuber architecture for its components suited to multi-agent intrinsic motivation based on representation and uncertainty.

2.2 Curiosity Learning

The notion that intelligent agents, be they humans, animals, or machines, are inherently curious and actively seek novel information has a long history. One of the earliest scientific indications of this phenomenon was provided in this study [8]. This study observed that infants tend to spend more time observing novel visual stimuli as opposed to familiar ones. This so-called novelty preference served as empirical evidence for an innate form of curiosity. Even in early developmental stages, humans exhibit an intrinsic drive to explore the unknown. This work laid the psychological groundwork for numerous subsequent theories of intrinsic motivation. However, this approach was confined to the psychological domain and did not yet extend to computational or algorithmic frameworks.

An early paradigmatic approach to the systematic study of curiosity did not consider it as a random or arbitrary behavior, but rather as the expression of an intrinsically motivated need for information acquisition. In an theoretical work, it was assumed that curiosity arises under conditions of moderate cognitive arousal: while low arousal is associated with boredom and high arousal with stress, a moderate level of cognitive stimulation, such as novelty or surprise, induces a heightened state of curiosity [3]. Within this research papers, key conceptual distinctions were introduced, particularly the differentiation between perceptual and epistemic curiosity. In addition, the concept of collative variables was established, stimulus properties such as novelty, complexity, or ambiguity, which modulate the level of arousal and promote exploratory behavior. These features are referred to as "collative" because they rely on relational evaluations e.g., in comparison to what is already known or expected. It was assumed that such stimuli elicit curiosity by increasing cognitive activation and thereby encourage exploratory engagement with the environment. This theoretical framework has also gained significance in the field of artificial intelligence (AI). Numerous contemporary

approaches in curiosity driven learning rely on similar principles, particularly the assumption that artificial agents are intrinsically motivated to actively explore surprising or hard to predict states. In this respect, early work can be interpreted as a conceptual precursor to current models of intrinsic motivation in the context of reinforcement learning.
A major step toward formalizing curiosity driven learning in artificial agents was taken in the early 1990s. A report from that time proposed that agents could be intrinsically rewarded for learning, specifically, for making progress in improving their predictive models of the world. Rather than relying solely on external rewards (such as scores in a game), agents would receive internal rewards for encountering states that enhance their understanding of their environment. This idea of curiosity as driven by learning progress laid the foundation for many modern frameworks in curiosity driven learning [22].
Building on earlier work, researchers in the 2000s expanded these ideas further, particularly within the field of developmental robotics. A study from that period developed formal frameworks for intrinsic motivation and demonstrated how robots could autonomously explore complex environments driven by curiosity, resembling the way children gradually acquire knowledge about the world [20].
With the rise of deep reinforcement learning, curiosity driven learning underwent significant advances. A key development during this period was the introduction of the concept of pseudo counts a method that allows agents to estimate the novelty of states in high dimensional environments, such as those found in Atari games. This approach provided a practical mechanism for implementing curiosity driven behavior in large scale settings [2].
A major breakthrough occurred in 2017 with the introduction of a curiosity driven deep reinforcement learning framework that translated the concept of intrinsic motivation into a functional learning system. In this approach, curiosity was defined as the prediction error of a forward dynamics model that is, how poorly the agent could predict the outcomes of its actions. High prediction error signaled an "interesting" or unfamiliar state, indicating potential for learning. This formulation proved not only conceptually elegant but also empirically effective. In environments with sparse or absent external rewards, curiosity driven agents significantly outperformed traditional ones [21].
Just a year later, a comprehensive study explored various curiosity driven strategies, including both traditional prediction error methods and the novel concept of Random Network Distillation (RND), in which curiosity emerges from discrepancies between outputs of randomly initialized neural networks. The findings demonstrated that curiosity is not only a theoretically idea but also a scalable and robust mechanism for exploration in large scale reinforcement learning tasks [4].
In parallel [25] provide a comprehensive framework that unifies psychological curiosity theories with their computational counterparts in AI. The study introduces a quantitative model of artificial curiosity based on psychological constructs such as the arousal mechanism and collative variables, including novelty, complexity, uncertainty, and incongruity. These variables are used to compute stimulus intensity, which is then mapped onto a bell shaped “curiosity preference function”, balancing the motivational effects of curiosity against aversion to overwhelming or trivial stimuli. In reinforcement learning, this framework translates into designing intrinsic reward signals that reflect prediction errors, state novelty, or information gain, thereby encouraging agents to explore underrepresented regions of the state space.
Moreover, the authors distinguish several types of curiosity driven methods in RL, such as uncertainty-based, novelty-based, and incongruity-based strategies, and analyze their strengths and limitations in tasks with sparse rewards, partial observability, or high dimensional input. This work helps bridge psychological theory and practical RL implementation, offering guidance for designing robust and interpretable intrinsic motivation systems in artificial agents.
Since then, curiosity driven learning has continued to evolve, being integrated into hierarchical reinforcement learning, combined with world models, and applied to open-ended environments such as Minecraft or BabyAI. Yet the core principle remains unchanged: a curious agent learns not merely from external rewards, but from internal signals of surprise and discovery.

2.3 Multi Agent Reinforcement Learning

TK

Many real-world problems cannot be solved by a single agent due to the need for interaction among multiple independent entities within the same environment. A suitable approach to handle such scenarios is the use of multi-agent systems (MAS). These are particularly relevant in the field of reinforcement learning (RL), commonly applied to environments whose complexity prevents the prior design of an optimal agent [6, 7].
An RL agent learns by interacting with its environment and receiving rewards, with the goal of maximizing these rewards. When deploying multiple agents, the complexity of the problem increases significantly, as the actions of one agent can influence the rewards received by others. Consequently, each learning agent must coordinate its actions with other agents, which is also influenced by whether the environment is cooperative or competitive [5, 27].
Fundamentally, RL problems are based on Markov Decision Processes (MDP). An MDP consists of a set of states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, a transition function $P(s' \mid s, a)$, and a reward function $R(s, a)$. The RL agent aims to find a

policy $\pi(a \mid s)$ that maximizes the expected cumulative reward. Within RL methods, a distinction is made between value-based and policy-based approaches. A common value-based approach is Q-learning, whose update rule is defined as follows:

$$\hat{Q}(s,a) \leftarrow (1-\alpha)\,\hat{Q}(s,a) + \alpha\big[r + \gamma \max_{a'} \hat{Q}(s',a')\big],$$

where $\hat{Q}(s,a)$ represents the estimated Q-value for taking action a in state s, α is the learning rate, and γ is the discount factor. The Q-value quantifies the expected sum of future rewards obtained by taking action a in the current state s and subsequently following an optimal policy [27, 6]. The objective is to iteratively approximate the optimal Q-function and derive the optimal policy accordingly.

In contrast, policy-based methods directly optimize the policy without explicitly using a value function. This is achieved by parameterizing the policy $\pi_\theta(a \mid s)$ and optimizing the parameters θ by updating them in the direction of the gradient, thus maximizing the expected cumulative reward [17]. A prominent example of this class is the Actor-Critic architecture, which combines the strengths of both value-based and policy-based approaches. The actor is responsible for selecting actions based on the current parameterized policy, while the critic estimates a value function to evaluate the quality of the selected actions. This feedback enables more informed and efficient policy updates. The policy is then improved by adjusting the parameters in the direction suggested by the critic's evaluation, this often results in faster convergence [15].

While these methods have shown great success in single-agent scenarios, real-world environments often require multiple agents to interact within a shared space. In such cases, the traditional MDP framework is extended to what is known as a Markov Game, or Stochastic Game [27]. This formalism models the dynamics of multiple agents by defining state transitions and individual reward functions that depend on the joint actions of all agents. Markov Games thus provide a principled foundation for studying both cooperative and competitive multi-agent settings. However, multi-agent learning introduces several unique challenges. One of the primary difficulties is the problem of *non-stationarity*. From the perspective of an individual agent, the environment appears to change over time, not due to inherent dynamics, but because other agents are simultaneously learning and adapting their policies. As a result, traditional RL algorithms that assume a stationary environment may fail to converge [27]. Another challenge is the *credit assignment problem*, especially in cooperative settings where multiple agents contribute to a shared outcome. It becomes non-trivial to determine how much each agent's action contributed to the received reward, which is essential for learning effective individual policies. Additionally, there is the need to address *scalability*, as the joint action space grows exponentially with the number of agents, making coordination increasingly difficult [7]. In competitive or mixed environments, agents may have opposing goals. This setting requires the development of policies that are robust against adversarial strategies.

Multi-agent learning algorithms can generally be categorized into cooperative and competitive approaches. In cooperative settings, all agents pursue a common goal and share a global reward signal. A key distinction within cooperative learning is whether coordination among agents is explicitly enforced or emerges implicitly [6]. Coordination-free methods aim to avoid direct coordination by assuming that agents act independently and learn from the shared reward signal. One simple approach is *Team Q-learning*, where a single centralized learner observes the full state and all joint actions, learning a global Q-function. In contrast, coordination-based methods aim to explicitly model and support coordinated behavior among agents. A prominent example is the *Joint Action Learner*, where each agent learns Q-values conditioned on the actions of others [6]. In a competitive setting, the minimax principle applies: one player seeks to maximize their reward while the other strives to minimize it. Building on this, the Minimax-Q algorithm updates its Q-function using the minimax value computed over the joint action space. This ensures that the resulting strategy remains robust even against the opponent's strongest possible counter-strategy [5].

3 Experiment

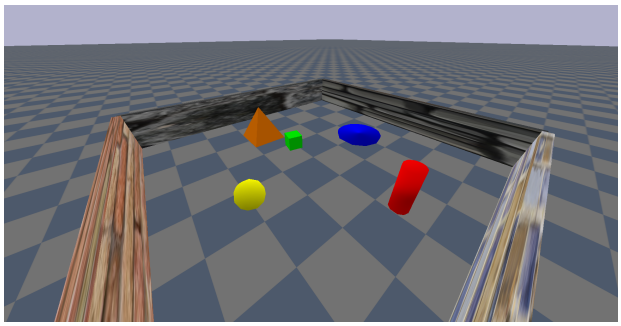
3.1 Environment

AT

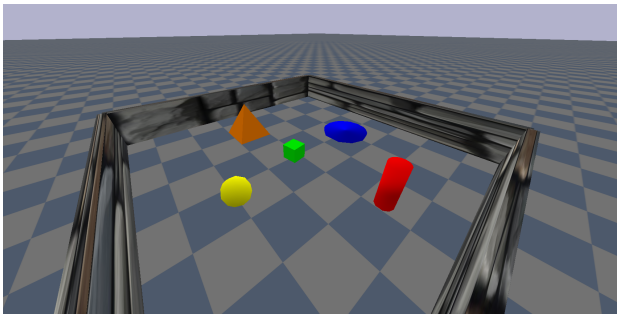
PyBullet is a lightweight physics simulation library based on the Bullet physics engine, particularly well-suited for robotics and reinforcement learning applications. With PyBullet, physically realistic simulations of rigid bodies, joints, collisions, and sensors can be easily implemented in Python. The library provides a simple API for defining and controlling robots as well as interacting with the simulated environment, making it ideal for use in learning-based control methods.

The environment used in this reinforcement learning project consists of a closed area with four surrounding walls. This area defines the boundary within which all agents and objects exist and interact. No agent or object can exit this bounded space. The floor of the environment is designed with a checkerboard pattern using alternating grey and

blue tiles. We have two different wall types. One is where all four walls have the same pattern, figure 1a, and the other is where each wall has its own pattern as you can see in figure 1b. We conducted experiments in both of these environments.



(a) Environment with different textures on each wall

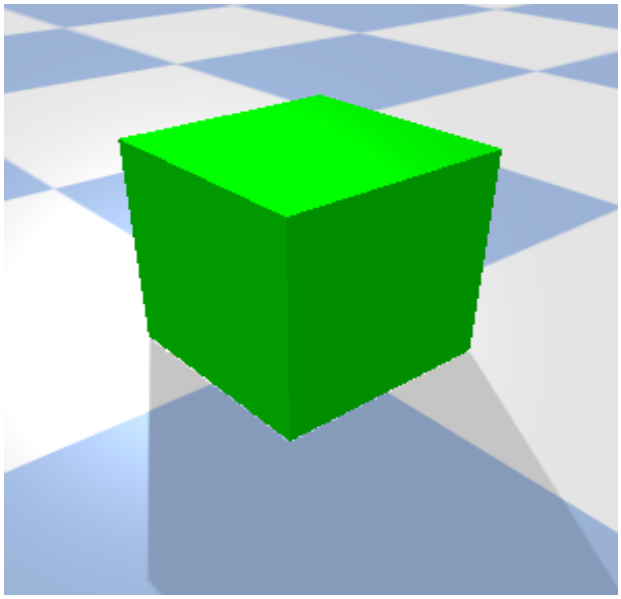


(b) Environment with identical textures on all walls

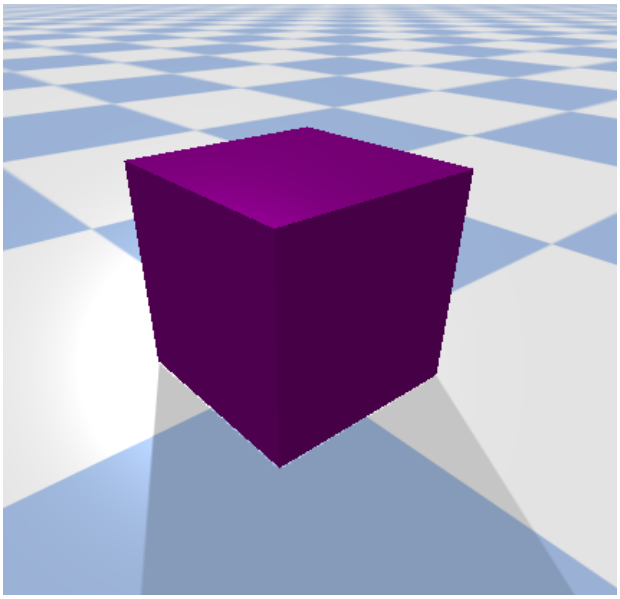
Figure 1: Environments

There are two agents in the environment, represented as colored cubes:

- Green Agent figure 2a
- Purple Agent figure2b



(a) First Agent



(b) Second Agent

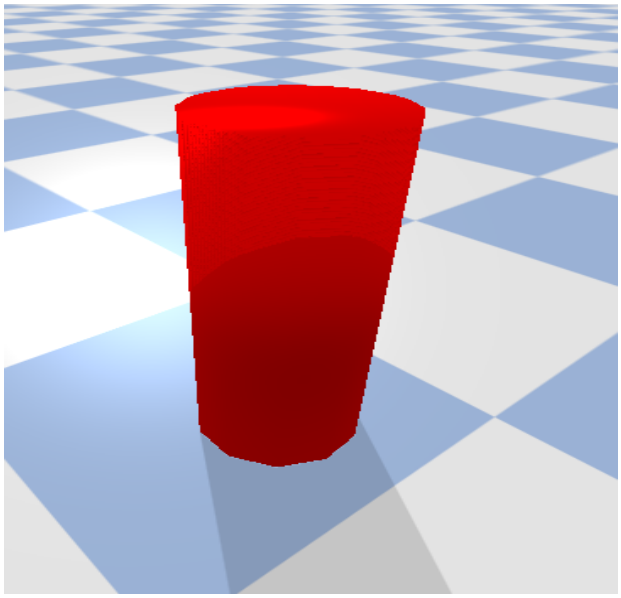
Figure 2: Representation of the two agents (purple and green).

The agents have the following set of actions available:

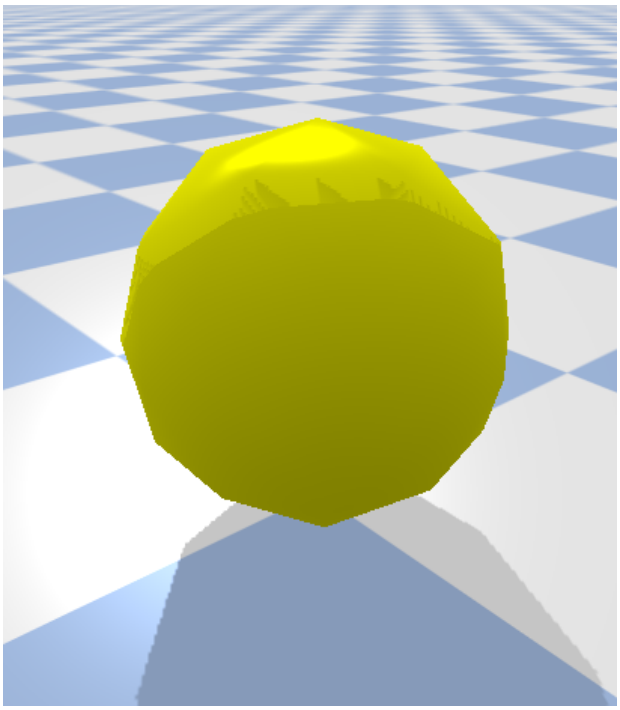
- Move forward
- Move backward
- Turn left
- Turn right
- Rotate left
- Rotate right

- Stop

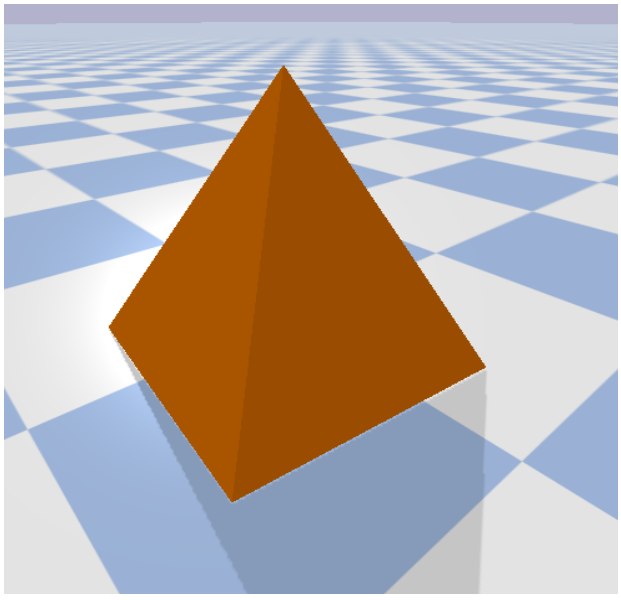
Agents can push objects by colliding with them.
In addition, the environment contains four static objects: an orange pyramid, a red cylinder, a yellow sphere and a blue disc. These objects cannot move on their own, but they can be pushed and displaced by the agents through physical contact.



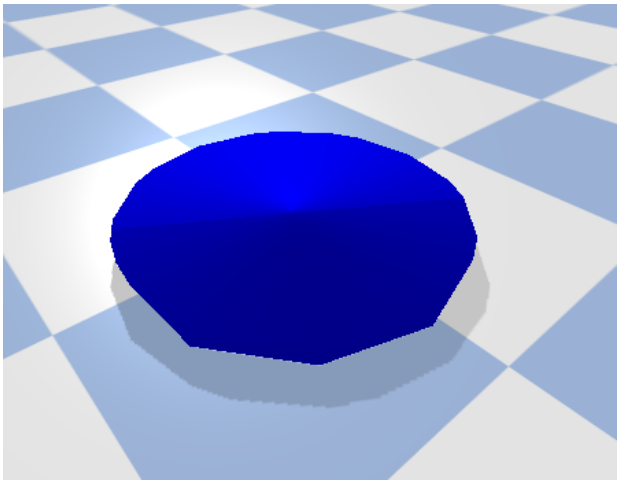
(a) Cylinder



(b) Sphere



(a) Pyramid



(b) Disk

3.2 Agent

The architecture of our agent, depicted in Figure 5, is designed to learn through intrinsic motivation driven by curiosity. It combines a predictive world model with a self-model that evaluates the predictive performance of the world model to encourage exploratory behavior. LS

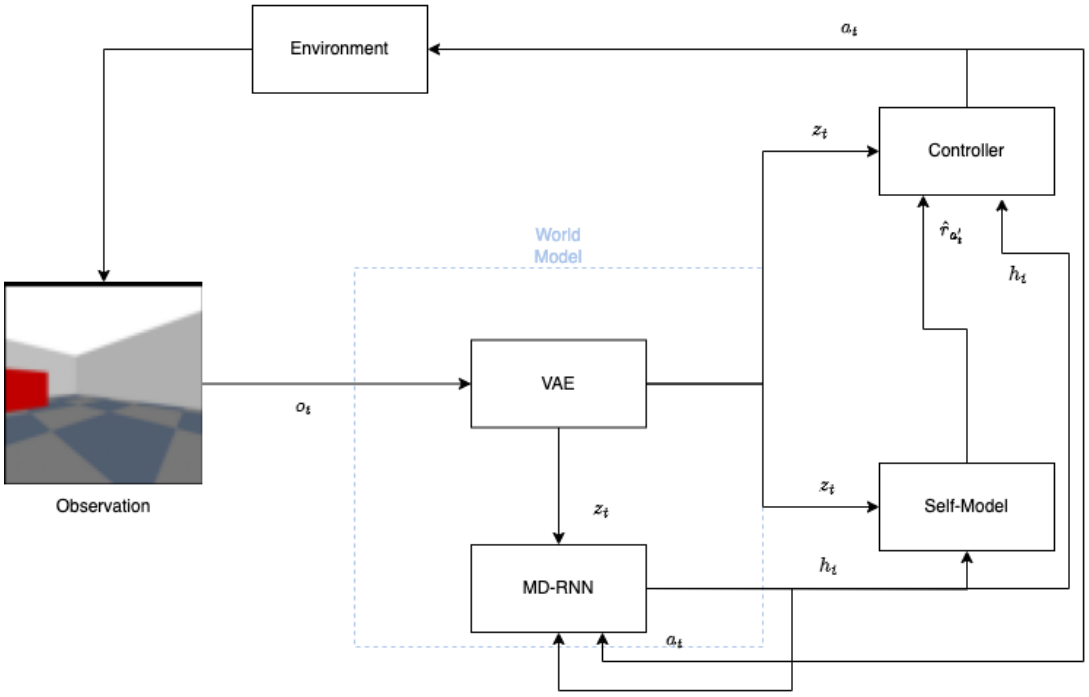


Figure 5: Architecture of the intrinsically motivated agent. The observation o_t from the environment is encoded into a latent state z_t by the VAE (V). The MDN-RNN (M) uses z_t and the action a_t to predict the next latent state z_{t+1} and update the hidden state h_t. The Self-Model (Λ) receives z_t and h_t and predicts the expected intrinsic reward $\hat{r}_{a'_t}$ for hypothetical actions a'_t. The Controller (C) uses z_t, h_t, and $\hat{r}_{a'_t}$ to select the next action a_t, which is sent to the environment. V and M together form the World Model.

Vision Model (VAE) The agent interacts with the environment by executing actions a_t and receiving observations o_t. The high-dimensional observation o_t (typically an image) is first compressed into a low-dimensional latent vector z_t by the Vision Model (V), implemented as a Variational Autoencoder (VAE) [14]. This step reduces the complexity for subsequent components and extracts relevant features from the visual input, similar to the approach by Ha & Schmidhuber [9]. The VAE is trained by minimizing the following loss:

$$L_{VAE} = \mathbb{E}_{q(z_t|o_t)}[\log p(o_t|z_t)] - \beta D_{KL}(q(z_t|o_t)||p(z_t)) \tag{1}$$

Here, the first term is the reconstruction loss (often implemented as Mean Squared Error, MSE, between o_t and the reconstruction $\hat{o}_t$) and the second term is the Kullback-Leibler divergence between the distribution $q(z_t|o_t)$ learned by the encoder and a prior distribution $p(z_t)$ (typically $\mathcal{N}(0, I)$), weighted by a factor β.

Memory Model (MDN-RNN) The latent vector z_t, along with the previous action a_t, serves as input to the Memory Model (M), a recurrent neural network (here, an MDN-RNN [9]). The Memory Model learns the temporal dynamics of the environment in the latent space and predicts the distribution of the next latent state $P(z_{t+1}|z_t, a_t, h_t)$, where h_t is the hidden state of the RNN. This hidden state h_t represents the agent's memory of the past interaction history. The MDN-RNN is trained by minimizing the negative log-likelihood (NLL) of the actual next latent states z_{t+1} under the predicted distribution:

$$L_M = -\log P(z_{t+1}|z_t, a_t, h_t) \tag{2}$$

Self-Model (Λ) and Curiosity In parallel, the current state, represented by z_t and h_t, is utilized by the Self-Model (Λ). Inspired by Haber et al. [10], the Self-Model predicts the expected error or intrinsic reward $\hat{r}_{a'_t}$ that the World Model (M) would incur if a hypothetical action a'_t were executed. The training target for the Self-Model is the actually computed intrinsic reward $r_{curiosity}$. This is often defined as the prediction error of the World Model (M), e.g., via the Mean Squared Error between the predicted next latent state $\hat{z}_{t+1} = \mathbb{E}[P(z_{t+1}|z_t, a_t, h_t)]$ and the actual next latent state z_{t+1}:

$$r_{curiosity} = ||z_{t+1} - \hat{z}_{t+1}||^2 \tag{3}$$

The Self-Model is then trained by minimizing the error between its prediction $\hat{r}_{a_t}$ (for the actually executed action a_t) and the target value $r_{curiosity}$, typically using MSE:

$$L_\Lambda = ||r_{curiosity} - \hat{r}_{a_t}||^2 \tag{4}$$

CONTROLLER / POLICY (C) The Controller (C) (or policy) receives the current latent state z_t, the memory state h_t, and the predictions $\hat{r}_{a'_t}$ from the Self-Model for various hypothetical actions. Based on this information, particularly the "surprise value" or "learning potential" of different actions quantified by the Self-Model, the controller selects the next action a_t to execute using an ϵ-Greedy strategy. With probability $1 - \epsilon$, the action maximizing the predicted reward (or error) from the Self-Model is chosen, and with probability ϵ, a random action is selected for exploration. This action a_t is then sent to the environment, influencing the next state and closing the learning loop. Maximizing the reward predicted by the Self-Model (which often correlates with the World Model's prediction error) drives the agent to seek novel and informative interactions.

3.2.1 POLICY

Epsilon-Greedy The ϵ-greedy policy [26] is a widely used strategy in reinforcement learning that provides a simple and intuitive way to balance exploitation and exploration. With a probability of $\epsilon \in [0, 1]$, the agent selects a random action from the set of all possible actions (exploration). With the remaining probability $1 - \varepsilon$, the agent chooses the action with the currently highest estimated value (exploitation). So we get AT

$$\pi(a \mid s) = \begin{cases} \text{a random action}, & \text{with probability } \epsilon \\ \arg\max_a Q(s, a), & \text{with probability } 1 - \epsilon \end{cases}$$

This approach allows the agent to occasionally try out new or rarely used actions, which can lead to discovering better strategies in the long run. At the same time, it mostly exploits its current knowledge by usually selecting the action that appears to be the best.
If the parameter ϵ is set to a high value, the behavior becomes more random. At the point $\epsilon = 1.0$, it is entirely random. In contrast, a very low value (e.g., $\epsilon = 0.01$) results in almost fully deterministic (greedy) behavior.

Boltzmann-Policy The Boltzmann policy [26] is a strategy in reinforcement learning that enables an agent to decide which action to take in a given state. Like other policies, it aims to balance exploitation and exploration.
The probabilities for action selection are calculated using a softmax function, which is based on the Q-values of the possible actions. The higher the Q-value of an action, the higher the probability that this action will be selected. The Q-value here represents the expected reward. The temperature parameter influences how strongly the differences between Q-values affect the resulting probabilities.
At high temperatures, the probabilities are more evenly distributed, which encourages the agent to explore more. At low temperatures, the probabilities are dominated by the Q-values, leading to almost deterministic behavior where the best action is chosen most of the time.
The formula for the Boltzmann strategy is

$$P(a_i) = \frac{e^{\frac{Q(a_i)}{T}}}{\sum_j e^{\frac{Q(a_j)}{T}}}.$$

Here, $Q(a_i)$ is the expected reward for action a_i, and T is the temperature. This formula ensures that actions with higher Q-values have exponentially higher selection probabilities, but still assigns non-zero probabilities to all actions, allowing for continued exploration.

Upper Confidence Bound The Upper Confidence Bound (UCB) [26] policy is a strategy primarily used in the context of multi-armed bandits and reinforcement learning to achieve a theoretically grounded balance between exploitation and exploration. Unlike ε-greedy or Boltzmann strategies, UCB does not rely on random exploration. Instead, it selects actions based on an upper confidence estimate of their potential reward.
The UCB algorithm estimates the average return of each action, while also taking into account the uncertainty of this estimate. At each time step t, the action a that maximizes the following expression is selected

$$a_t = \arg\max_a \left[\hat{Q}(a) + c \cdot \sqrt{\frac{\ln t}{N(a)}} \right].$$

here $\hat{Q}(a)$ is the estimated average reward of action a, $N(a)$ is the number of times action a has been selected so far, t is the total number of time steps so far and $c > 0$ is a constant that controls the degree of exploration. A higher value leads to more exploration, as less secure actions are more favored. A lower value of c therefore leads to more exploitation.
The term $\sqrt{\frac{\ln t}{N(a)}}$ represents the uncertainty or confidence interval of the estimated reward. Actions that have been selected less frequently receive a higher uncertainty value, encouraging the agent to explore them. As more observations are gathered, the confidence interval narrows, and the algorithm increasingly shifts toward exploiting the best-known actions.

3.3 Multi Agent

To extend the concept of curiosity-driven learning to a multi-agent context, we have developed a novel architecture LS
that integrates a shared predictive world model with individual, agent-specific motivational systems. As depicted in Figure 6, our framework supports two agents operating concurrently in the same environment. While the agents learn a unified model of the world's dynamics together, their exploratory actions are driven by independent self-models. This design is inspired by the world model concept of Ha and Schmidhuber [9] and the intrinsically-motivated self-aware agent framework proposed by Haber et al. [10]

Action Execution The simulation proceeds in synchronous steps. At each timestep t, the current observation for both agents (o_t^1, o_t^2) is captured. Based on this, each agent's controller independently and simultaneously selects an action (a_t^1, a_t^2). Both selected actions are then applied to the environment at the same time, and the physics simulation is advanced by one step to produce the next state. This parallel execution model ensures that the agents' decisions are based on the same world state and that their actions influence each other within the same timestep.

3.3.1 Shared World Model

The foundation of our architecture is a single world model that is shared between both agents. This model is responsible for learning a compressed spatial and temporal representation of the environment from the agents' combined experiences . It is trained in an unsupervised manner using data collected from both agents as they explore. The world model consists of two main components: a Vision Model (V) and a Memory Model (M) .

Vision Model (V) The Vision Model is implemented as a single Variational Autoencoder (VAE) that processes the high-dimensional observations (o_t) from both agents. At each timestep, the VAE's encoder compresses the raw pixel image from each agent into a low-dimensional latent vector $z_t^i \sim q(z_t^i|o_t^i)$. The shared VAE is trained by minimizing the aggregate loss over both agents, which consists of the reconstruction error and a Kullback-Leibler (KL) divergence term to regularize the latent space :

$$L_{VAE} = \sum_{i=1}^{2} \left(\mathbb{E}_{q(z_t^i|o_t^i)}[\log p(o_t^i|z_t^i)] - \beta D_{KL}(q(z_t^i|o_t^i)||p(z_t^i)) \right) \tag{5}$$

Here, the first term is the reconstruction log-likelihood, and the second term regularizes the posterior $q(z_t^i|o_t^i)$ to be close to a prior $p(z_t^i)$, typically a standard normal distribution $\mathcal{N}(0, I)$.

Memory Model (M) The Memory Model's role is to learn and predict the temporal dynamics of the environment within the latent space created by the VAE. It is implemented as a Mixture Density Network-Recurrent Neural Network (MDN-RNN) that models the probability distribution of the next latent state, $P(z_{t+1}|z_t, a_t, h_t)$, where h_t is the RNN's hidden state . In our multi-agent setting, the MDN-RNN receives the actions a_t^1, a_t^2 and latent vectors z_t^1, z_t^2 from both agents. Its hidden state h_t thereby serves as a shared memory, integrating historical information from both agents to predict future states . The model is trained by minimizing the negative log-likelihood (NLL) of the actual next latent states under the predicted distribution for both agents :

$$L_M = -\sum_{i=1}^{2} \log P(z_{t+1}^i|z_t^1, z_t^2, a_t^1, a_t^2, h_t) \tag{6}$$

3.3.2 Individual Self-Models (A) and Controllers (C)

While the world model is shared, the motivation for exploration is generated individually for each agent. This is achieved through a combination of a private Self-Model (A) and a Controller (C).

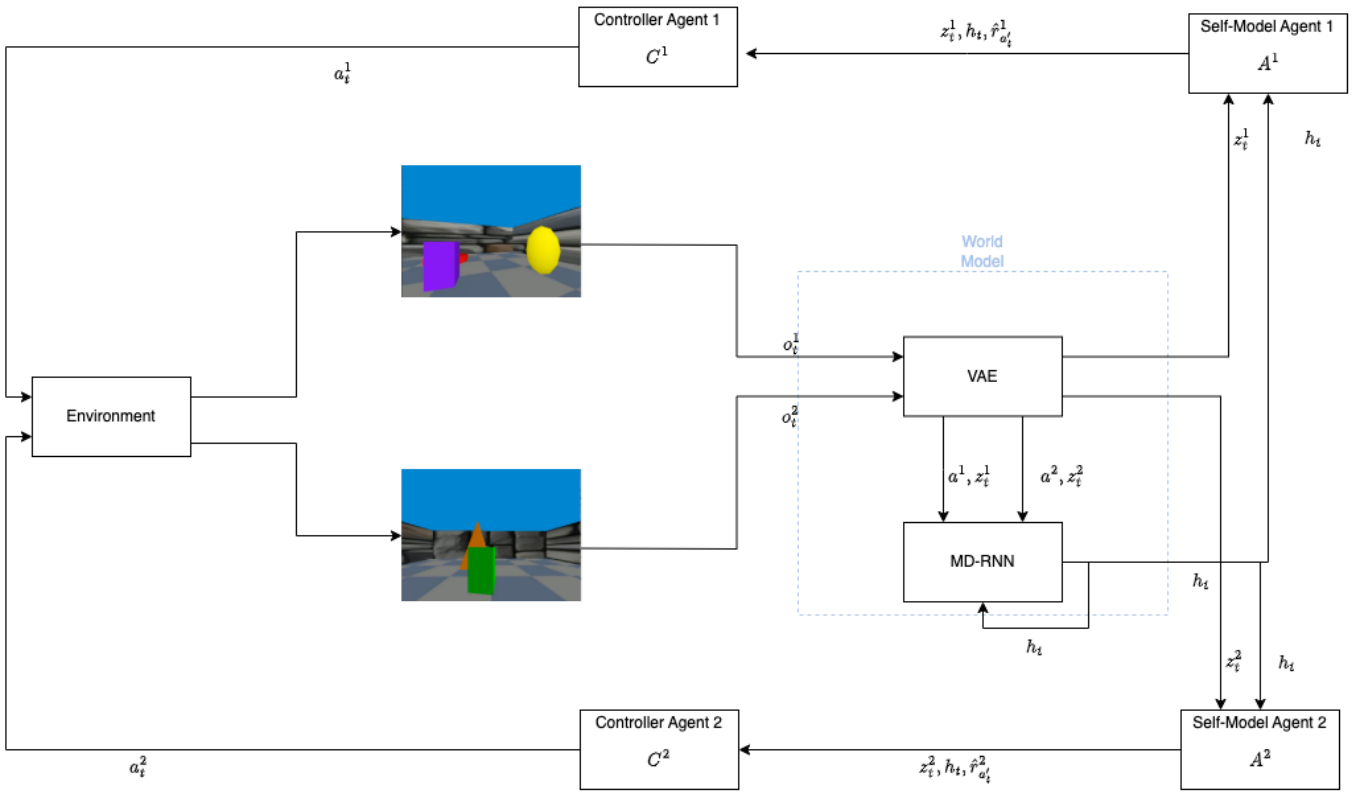


Figure 6: The proposed multi-agent architecture for curiosity-driven learning. Two agents interact with a shared environment, each receiving a unique observation (o_t^i). A **Shared World Model**, composed of a Variational Autoencoder (VAE) and a Mixture Density Network RNN (MDN-RNN), processes these inputs. The VAE (V) compresses each high-dimensional observation o_t^i into a low-dimensional latent vector z_t^i. The MDN-RNN (M) receives the latent vectors and actions (a_t^1, a_t^2) from both agents to model the environment's temporal dynamics, updating a shared hidden state h_t. Each agent possesses an Individual Self-Model (A^i) which uses the agent's latent state z_t^i and the shared hidden state h_t to predict the intrinsic curiosity reward ($\hat{r}_{a'_t}^i$) for hypothetical actions. This reward signal, along with the state representations z_t^i and h_t, is fed into the agent's Individual Controller (C^i). The controller selects the next action a_t^i to maximize this intrinsic reward, promoting novel and exploratory behavior.

Self-Model (A^i) for each agent learns to predict the error of the shared world model . It takes the agent's own latent state z_t^i and the shared hidden state h_t as input and, for a set of hypothetical future actions a'_t, predicts the expected intrinsic reward $\hat{r}_{a'_t}^i$. This reward corresponds to the world model's anticipated prediction error, effectively quantifying how "surprising" or "interesting" an action would be . Actions that are predicted to cause high error are considered more curious.

Controller (C^i) is the agent's policy network. It receives the current latent state z_t^i, the shared memory state h_t, and the intrinsic reward predictions $\hat{r}_{a'_t}^i$ from its self-model. The controller's objective is to select the action a_t^i that maximizes the predicted curiosity reward. For our multi-agent experiments, each controller utilizes the same policy framework as detailed for the single-agent case in Section 3.2.1.

4 Results

4.1 Single Agent

TK

The progressive enhancement of the variational autoencoder (VAE) employed in our system over the course of training is shown in Figure 7. In the upper panel, after merely 500 training steps, the broad scene layout, comprising sky, ground and principal objects, is discernible, but the reconstructions remain highly blurred, with the exact colors and shapes of the cylinder, sphere, disk and pyramid largely obliterated. By contrast, in the lower panel, after approximately 79 000 steps, the model achieves near-faithful reproduction: sky gradients are crisply defined, ground

textures appear convincingly realistic, and the geometric contours of each object closely mirror those of the input images. This substantial improvement in reconstruction fidelity confirms that the VAE furnishes a robust basis for precise state prediction in the world model and for the derivation of informative intrinsic rewards in the self-model.

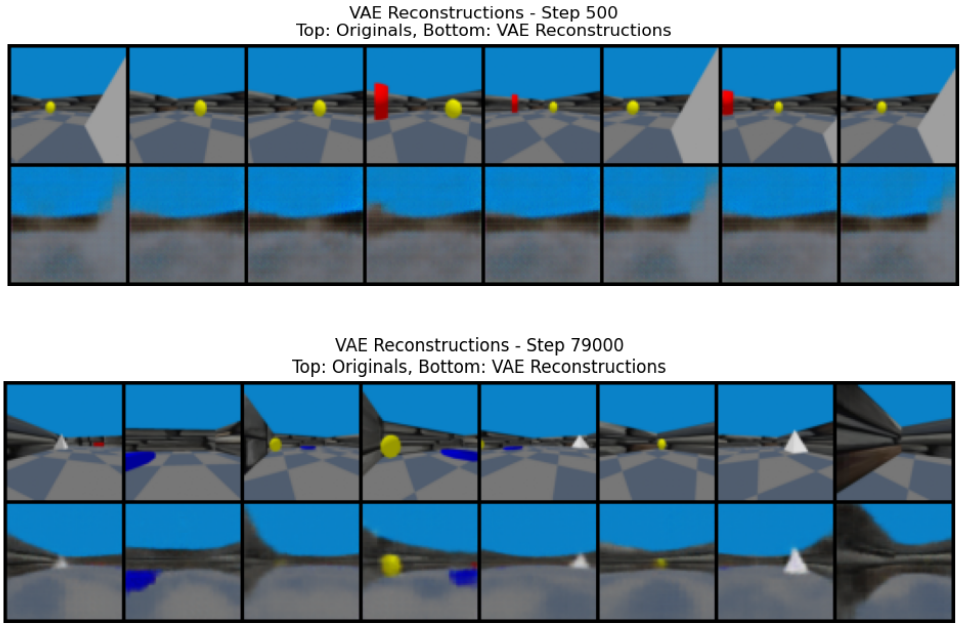


Figure 7: Comparison of VAE reconstructions at two training steps. Top: Step 500; Bottom: Step 79,000.

Figure 8 illustrates the evolution of the intrinsic curiosity reward over time for a curiosity-driven agent in the environment. In the initial phase the curiosity reward hovers at a relatively high level close to 1, indicating that the agent is encountering unfamiliar states in which the RNN model makes large prediction errors. By around step 5000 the curiosity reward exhibits a pronounced decline to approximately 0.55. This drop corresponds to a period of intensive model adaptation during which the RNN progressively learns to predict the environment's latent state more accurately. As a result, the intrinsic reward decreases significantly, reflecting the reduction in prediction errors. After step 10 000 the reward stabilizes in the range of 0.57 - 0.59. From step 15 000 onward the smoothed curve reveals a gradual but consistent increase in the curiosity reward, rising from about 0.57 to roughly 0.63 by the end of the simulation at step 80 000. This slow ascent can be attributed to the agent's exploration of increasingly complex or less frequently visited states, where the RNN again incurs slightly larger prediction errors. Overall, this trend suggests that, while the model learns robust latent embeddings over time, the agent continues to discover new situations that moderately elevate the intrinsic reward. Notably, the model appears to have effectively converged by around step 40 000. The pronounced fluctuations in the raw data stem from discrete changes in the observed scene such as collisions or changes in direction. Applying smoothing renders the underlying trend more transparent.

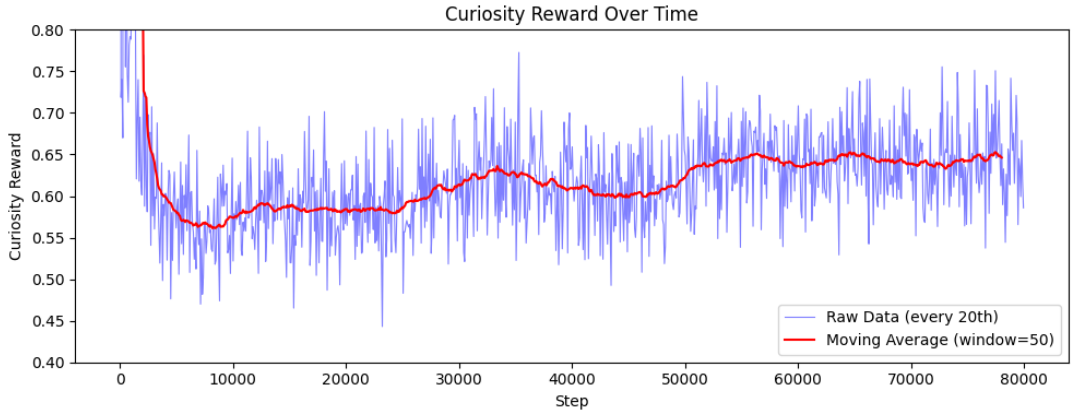


Figure 8: Evolution of the intrinsic “curiosity reward” over the simulation steps. The blue line shows the raw data sampled every 20 steps, while the red line represents the moving average over a window of 50 points.

The evolution of curiosity is further corroborated by the agent's object-interaction patterns shown in Figure 9. Around 25 000 training steps, coinciding with the steepest ascent in curiosity reward, the agent begins to interact frequently with objects for the first time. Video recordings indicate that at this point the agent actively "plays" with the sphere in a manner reminiscent of kicking a soccer ball. Furthermore, the agent exhibits a marked preference for objects capable of motion, such as the sphere and the toppled rolling cylinder, while rapidly losing interest in the remaining static objects after only brief contact. The world model is able to predict these ongoing changes with considerable accuracy over time, nevertheless, a persistent prediction error remains, punctuated by regular brief peaks in which forecast quality temporarily degrades.

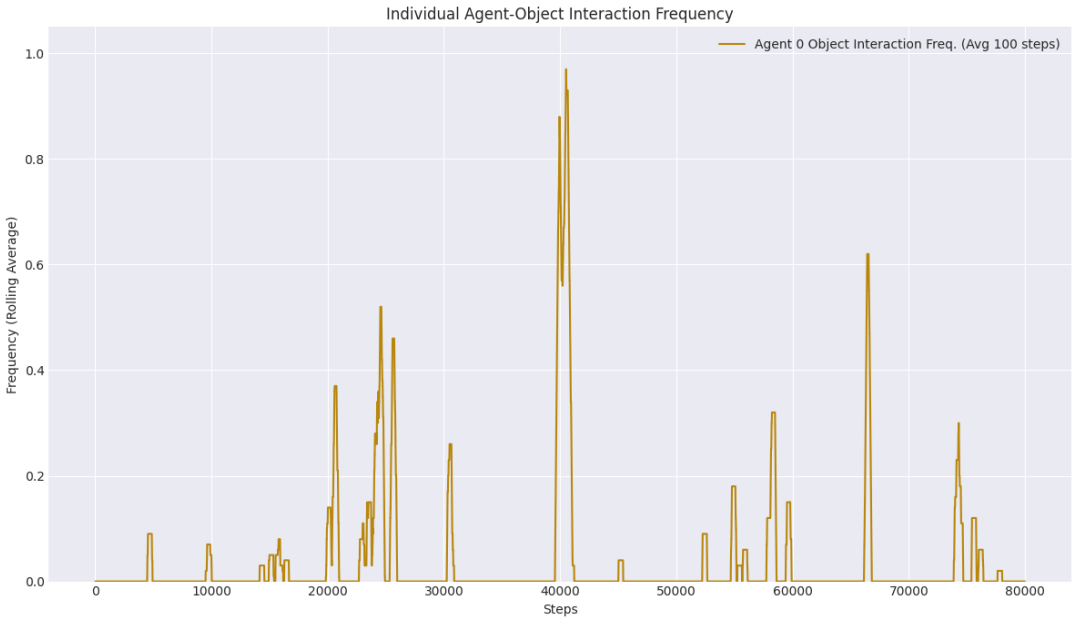


Figure 9: Frequency of agent-object interaction, averaged over 100 steps.

Furthermore, over the course of training the agent learns which actions are particularly relevant to its task, as illustrated in Figure 10. In this run, an ϵ-greedy policy with $\epsilon = 0.2$ was employed. The "stop" action simply causes the agent to remain in place for one time step and generate no new observation. Because, in this case, the world model can predict the previous state almost perfectly, the stop action incurs a very low prediction error and thus yields a correspondingly low intrinsic reward. Consequently, the agent selects this action only rarely, as clearly reflected in the resulting action distribution.

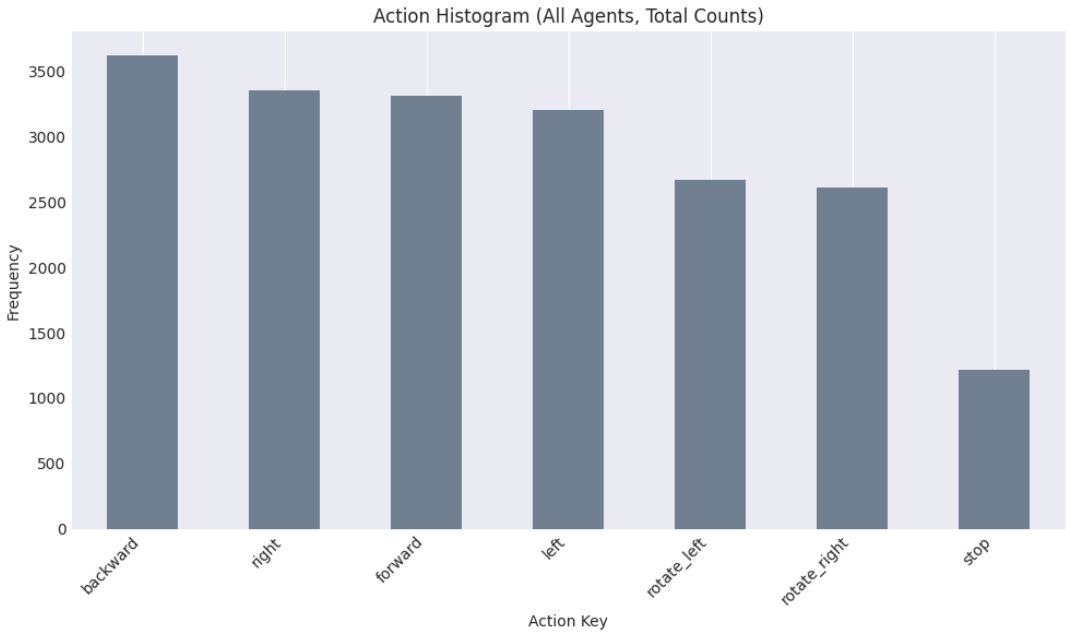


Figure 10: Histogram showing the distribution of action counts for all agents over the entire simulation period.

During simulation experiments with different exploration policies, the Boltzmann strategy produced by far the lowest frequency of agent-object interactions compared to ϵ-greedy and UCB. This deficiency is clearly reflected in the intrinsic curiosity rewards shown in Figure 11. The ϵ-greedy and UCB curves follow very similar trajectories, reaching moderate to high reward levels over time, while the Boltzmann policy consistently underperforms and records substantially lower curiosity rewards throughout the simulation.

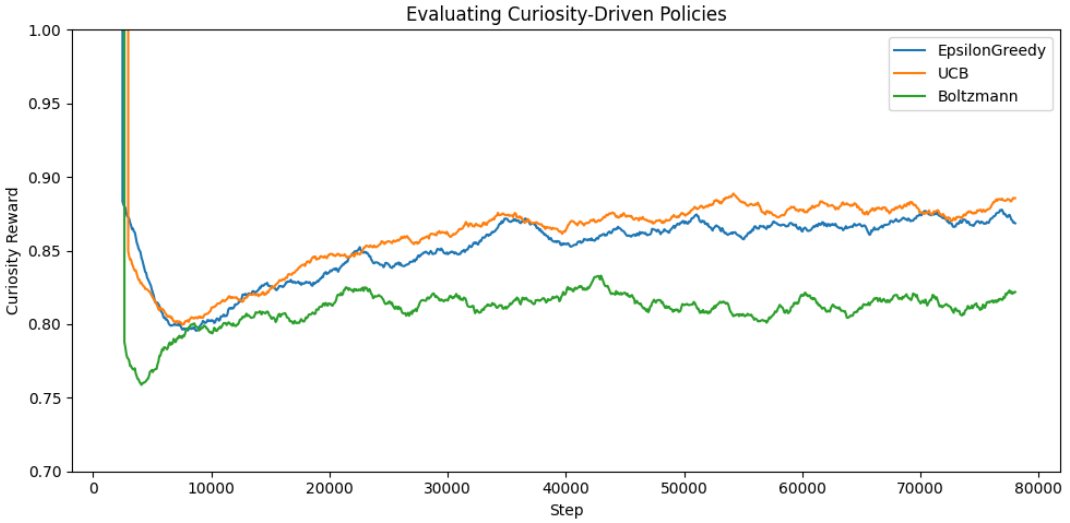


Figure 11: Evolution of the intrinsic curiosity reward over simulation steps under three exploration strategies: ϵ-greedy, UCB and Boltzmann.

4.1.1 All walls different

In this experiment, all four walls were assigned different patterns. Furthermore, the ϵ-greedy policy with $\epsilon = 0.2$ was employed in a single-agent environment. In Figure 12, a histogram of all actions chosen by the agent is presented. Similar to previous experiments, the action `stop` was selected least frequently. This is due to the agent receiving minimal curiosity loss for this action, resulting in a lack of intrinsic motivation to explore it. The actions `forward`, `left`, `right`, and `backward` exhibit an approximately uniform distribution, indicating no strong bias toward any

particular movement direction. The actions `rotate_left` and `rotate_right` were selected somewhat less frequently, suggesting a lower exploratory value attributed to rotational behavior in this setting.

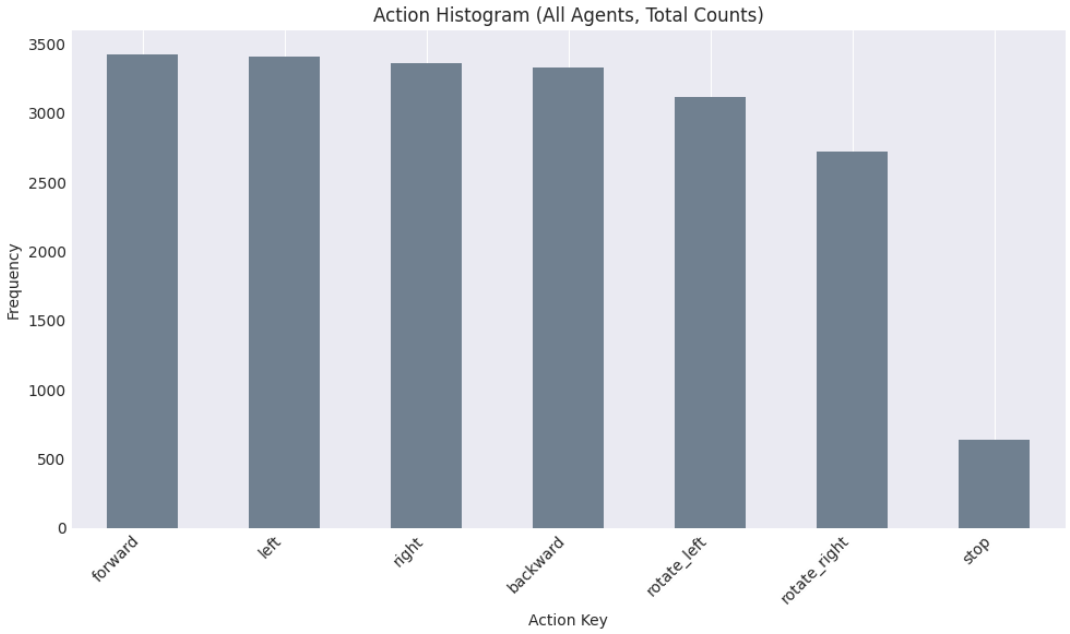


Figure 12: Histogram showing the distribution of action counts for all agents over the entire simulation period.

Furthermore, we also analyzed the VAE reconstructions at training steps 5,000 and 75,000. As shown in Figure 13, the images become noticeably sharper over time. The objects exhibit correct colors and well-defined contours. In addition, objects located farther from the agent are more clearly distinguishable in the later stage of training. For instance, compare columns 3 and 6 in the lower panel with columns 2 and 5 in the upper panel.
Moreover, the model successfully learned to reconstruct all four wall textures, despite their considerable variation—from black-and-white patterns to natural wood and wood interlaced with blue elements. In the upper panel (step 5,000), the walls are only approximately represented, with color tones loosely matching the originals but lacking any structural pattern. By contrast, after 75,000 training steps, the model can clearly differentiate between the wall textures and reproduce them with much greater fidelity. For example, in column 5 of the lower panel, the wooden texture with distinctive blue accents is clearly visible. The same wall appears in columns 1 and 5 of the upper panel, where the blue elements are not discernible, indicating the model's earlier inability to capture fine-grained texture details.

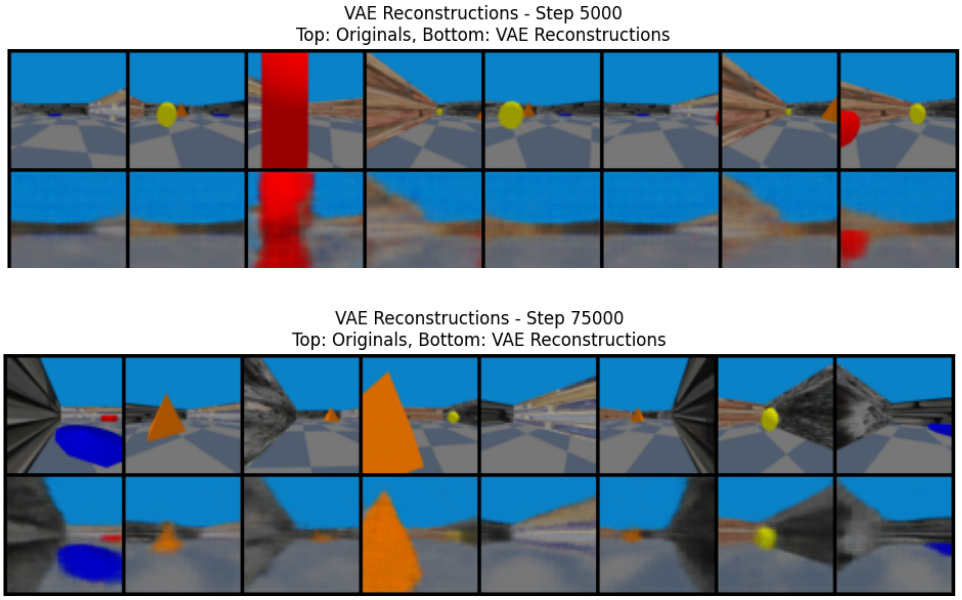


Figure 13: Comparison of VAE reconstructions at two training steps. Top: Step 500; Bottom: Step 79,000.

Figure 14 illustrates the agent's interaction with four different objects during the training process. As seen in the accompanying video, the agent initially shoots the disk. It subsequently locates the cylinder and knocks it over, enabling it to roll. The agent continues to find this object of interest until approximately step 5000.
Between steps 14000 and 38000, the agent primarily interacts with the disk and the pyramid. It pushes the disk close to the pyramid early in this phase and then engages with both objects repeatedly. Toward the end of this interval, the disk is pushed across the area towards the cylinder.
The sphere is also located near the cylinder and the disk, but at a slightly greater distance. While the agent occasionally shows interest in the sphere, the interactions are less frequent and less intensive compared to those with the other objects. The highest level of interaction is observed with the sphere; however, since it is situated in a corner, it cannot roll far.
Subsequently, the agent moves to the corner where the cylinder and the disk are located and makes a final attempt to push them. Due to their constrained position in the corner, their movement is significantly limited.

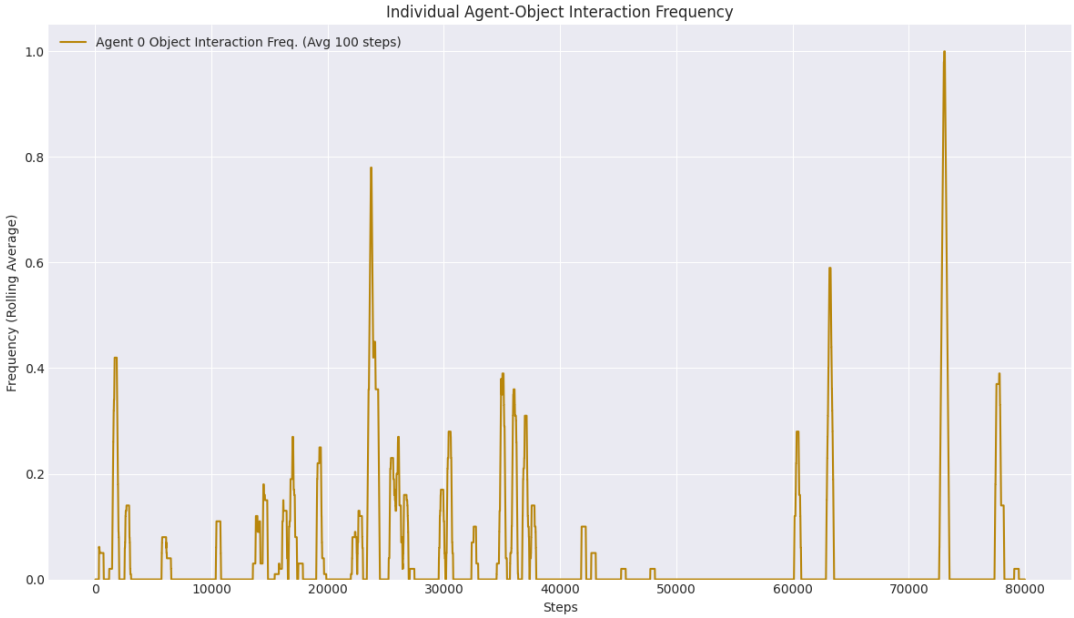


Figure 14: Frequency of agent-object interaction, averaged over 100 steps.

Figure 15 illustrates the evolution of the intrinsic curiosity reward over time for a curiosity-driven agent within the environment. At the beginning of training, the recurrent neural network (RNN) model is still untrained, leading to high prediction errors. As learning progresses, the model improves rapidly, resulting in a sharp decline in the curiosity reward. Between 10,000 and 40,000 steps, the model still exhibits fluctuations of up to 0.08. After this phase, a clear stabilization of the system can be observed, with the curiosity reward consistently hovering around a value of 0.60 from step 40,000 onward. This indicates that the curiosity reward has converged.

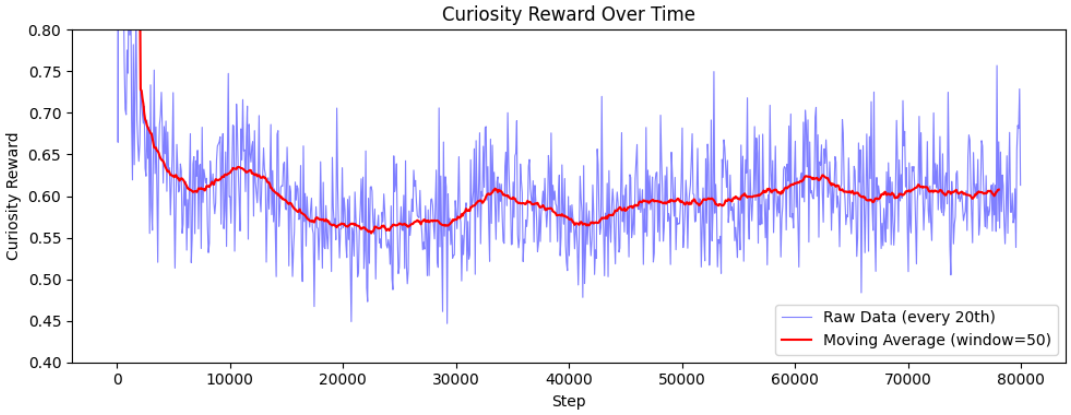


Figure 15: Evolution of the intrinsic "curiosity reward" over the simulation steps. The blue line shows the raw data sampled every 20 steps, while the red line represents the moving average over a window of 50 points.

4.1.2 Multi-Agent ϵ-Greedy

For each agent, a video was recorded to enable detailed analysis of their individual actions within the scenario. The videos were saved in reduced quality to conserve storage space. Nevertheless, it remains possible to clearly observe the agents' behaviors and their interactions with the environment. Figure 16 shows a segment from one of these recordings, taken from the perspective of the second agent. The image on the left depicts the first agent positioned behind a fallen cylinder. The image on the right captures the subsequent moment, in which the first agent has rolled the cylinder toward the wall, while the second agent observes attentively. This situation continues for a short time until the second agent eventually turns away. This example illustrates that the agent tends to focus on situations whose outcomes it cannot fully predict. It exhibits exploratory behavior and directs its attention toward the unfolding interaction. As shown in 17, the curiosity reward for both agents can also be observed in the step range between 45,000 and 47,000. In this situation, the curiosity reward is notably high, reaching just below 0.60. A high curiosity reward is of particular interest, which explains why Agent 2 engages with this scenario. This scenario can also be seen in the plot 21. It shows that the agent-to-object interaction increased between the 45,000 and 50,000 steps.

TK
LS
AT

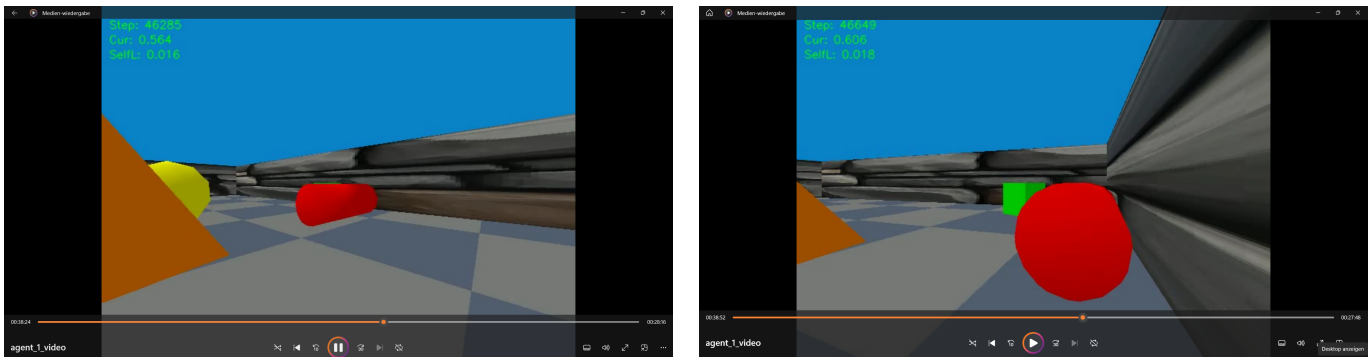


Figure 16: Agent 2: left: Start of the situation, Right: End of the situation

As described in the multi-agent experiment utilizing an ϵ-Greedy, Figure 17 shows the moving average of the curiosity reward for each of the two agents The reward trajectories for both Agent 0 and Agent 1 are remarkably similar, following nearly identical trends throughout the experiment. Both agents start with a high curiosity reward of 0.7, which steeply declines to a low of approximately 0.53 between 20,000 and 25,000 steps. This initial drop signifies a

rapid learning phase where the shared world model quickly adapts to the environment's basic dynamics based on the agents' initial explorations.
Following this initial phase, the curiosity rewards for both agents enter a gradual and synchronized ascent, reaching a value over 0.6 by the simulation's end. This shared upward trend suggests the agents are collectively exploring more complex behaviors, such as interacting with dynamic objects. Such dynamic interactions are harder for the world model to predict accurately, resulting in higher prediction errors and, consequently, a greater curiosity reward. As can also be seen in Figure 17, the model has not yet reached a stable state. The curiosity reward still fluctuates significantly, indicating that the environment has not been fully learned. This suggests that there is still potential for further improvement, and it may be beneficial to conduct a longer training run of up to 200,000 steps.

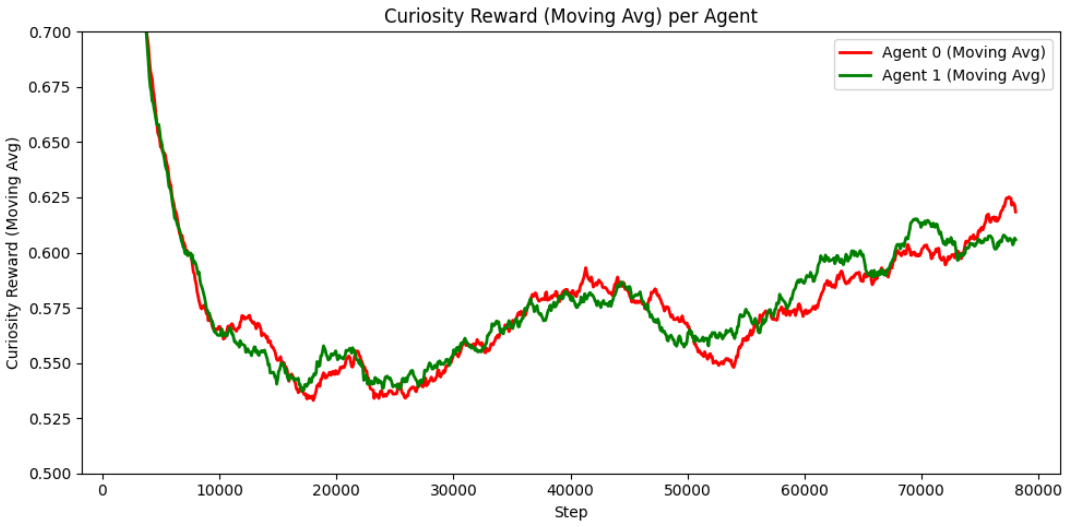


Figure 17: Curiosity Reward Multi-agent ϵ-Greedy

4.1.3 Multi-Agent UCB

In Figure 18, the curiosity reward is shown separately for each agent for the UCB policy with $c = 1$. Initially, it starts at a value greater than 1 and then decreases rapidly to below 1. It reaches its minimum between steps 10.000 and 30.000. After this period, the curiosity reward continues to exhibit some fluctuations, but overall it increases significantly. Beyond this range, it typically remains between 0.65 and 0.70. Its maximum value is slightly above 0.70. Figure 21 illustrates the interaction between the agents and the objects. It can be observed that the agents interact extensively with the objects overall. Of all the policies, this one has the highest interaction with the objects. This is one reason why the curiosity reward is comparatively high. Furthermore, it is noticeable that the two agents can receive significantly different curiosity rewards. For example, at around 27.000 steps and 47.000 steps, there is a curiosity reward difference of approximately 0.05 and 0.03, respectively. Furthermore, Figure 22 shows the agent-to-agent interaction, which is also high when using the UCB policy. This indicates that the agents interact frequently both with each other and with the objects in the environment. Such frequent interactions lead to a constantly changing environment, which in turn triggers a high level of curiosity in the agents and results in an increased curiosity reward.

TK
LS
AT

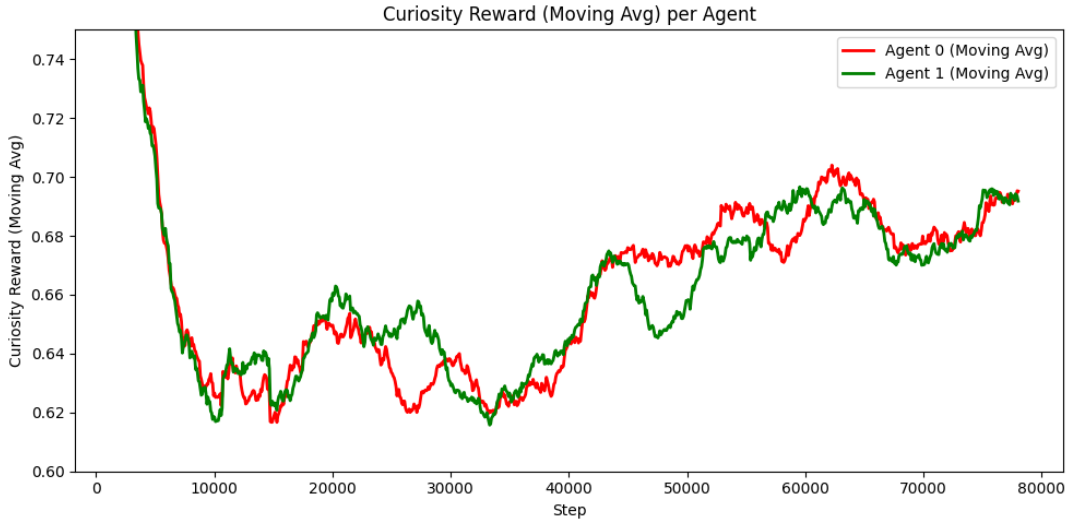


Figure 18: Curiosity Reward Multi-agent UCB

4.1.4 Multi-Agent Boltzmann

To find an optimal exploration strategy, a series of experiments was conducted using a Boltzmann policy with different temperature settings. We tested fixed temperature values of 0.5, 1.0, 1.5, 2.0, and 2.5. As shown in Figure 19, none of the tested fixed-temperature configurations proved to be conclusively superior. All settings resulted in similar performance trajectories, with the curiosity reward converging to a comparable range after an initial drop. This suggests that a static temperature value within this range does not significantly alter the agent's learning behavior or final performance.

TK
LS
AT

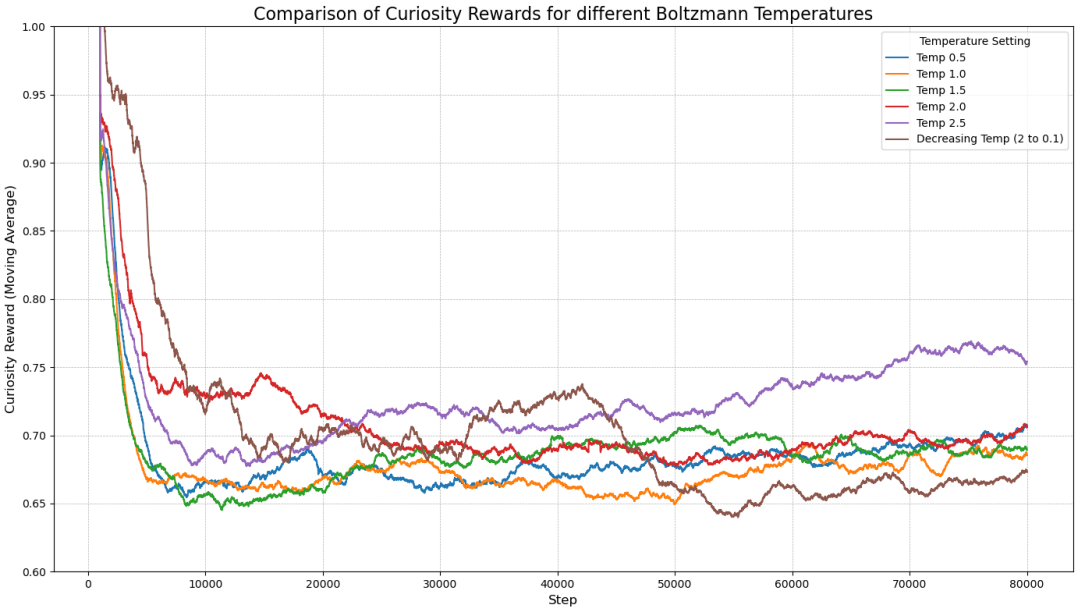


Figure 19: Compare different Temperatures

The curiosity reward for each agent over the course of the training is depicted in Figure 20. Initially, between steps 15,000 and 35,000, a period characterized by agent-object interactions, the curiosity reward stabilized around a value of 0.7.
A significant event occurs around step 42,000, where the reward for both agents peaks. This corresponds to the phase where objects in the environment began to move, introducing new dynamics for the agents to learn and leading to a temporary increase in curiosity.

Following this peak, the reward experiences a sharp decline, reaching its lowest point at approximately step 55,000. This drop suggests a period of reduced novel interactions. Subsequently, the system transitioned into a phase dominated by agent-to-agent interactions. This new source of complexity and novelty caused the curiosity reward to steadily increase again towards the end of the experiment, indicating that the agents were actively learning from each other's behavior.

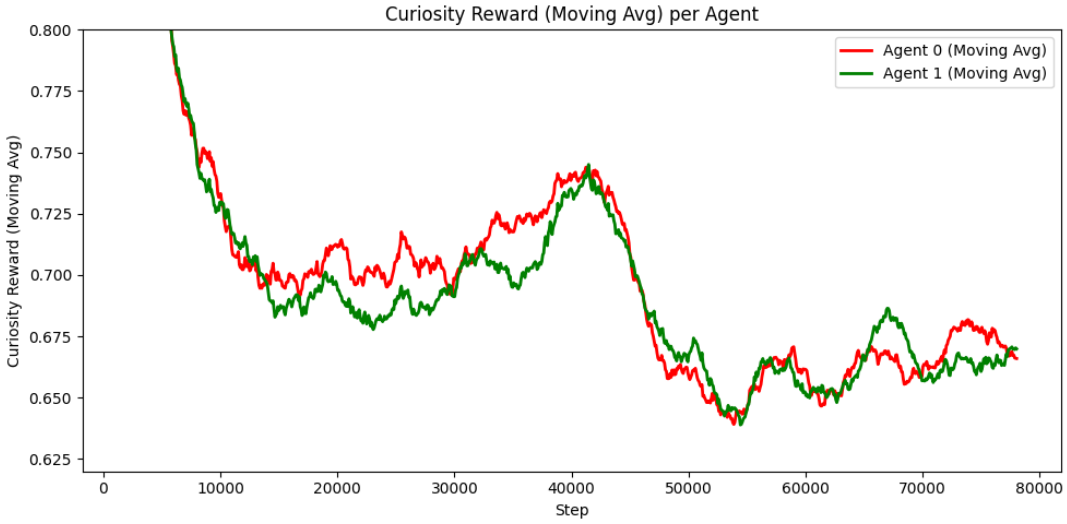


Figure 20: Curiosity Reward Boltzmann

4.1.5 Multi-Agent comparisons across all policies

In Figure 21, we present the interactions between the agents and the objects for all policies. For this purpose, the interactions of both agents were summed and analyzed per policy. The plot shows this total as a cumulative scenario. The global maximum number of interactions is normalized to 100% and is reached at 80,000 steps by the UCB policy. The orange line thus indicates that UCB achieves 60% of its total interactions by 40,000 steps. The blue line represents the progression for the ϵ-greedy policy, showing that after 80,000 steps, this policy achieved only about half as many interactions as UCB. The green line shows the progression for the Boltzmann policy. It can be observed that there were no further object interactions after 30,000 steps and that, in total, this policy reached only approximately 15% of the number of interactions achieved by UCB.

TK
LS
AT

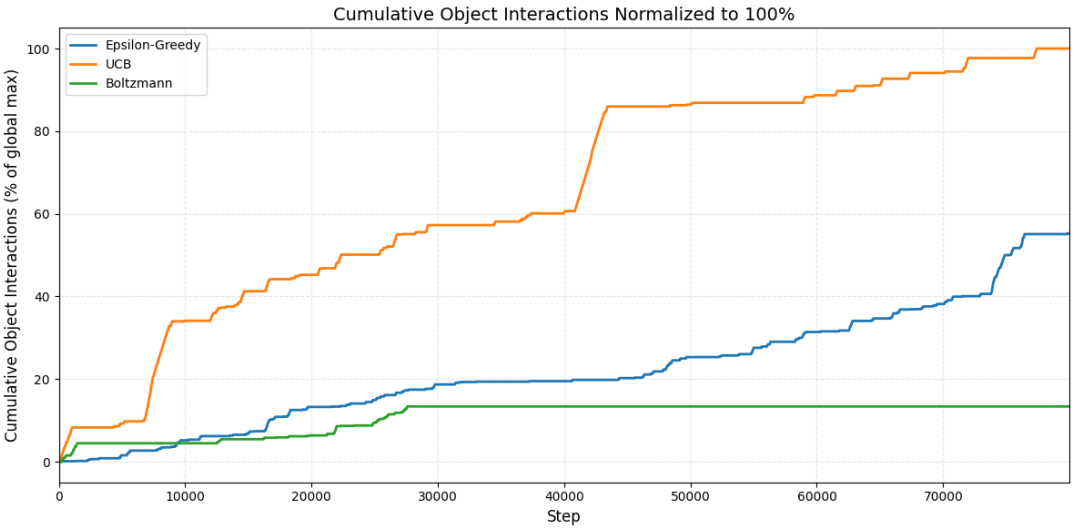


Figure 21: Cumulative Object Interactions per policy

In the multi-agent scenario, an agent can interact with objects, but it can also collide with the other agent, resulting in mutual displacement. These agent-to-agent interactions are shown in the plot in Figure 22. As before, a cumulative

global maximum value is used as a reference point. In the plot, the orange line represents the UCB policy. It is evident that UCB results in the highest frequency of agent-to-agent interactions. The blue line shows the results for the ϵ-greedy policy, which leads to almost no agent-to-agent interaction. After 80,000 steps, it reaches only about 10% of the global maximum. The green line, corresponding to the Boltzmann policy, shows a moderate level of agent-to-agent interaction, achieving roughly 50% of the maximum. Overall, UCB clearly outperforms both alternative policies in terms of agent-to-agent interaction frequency.

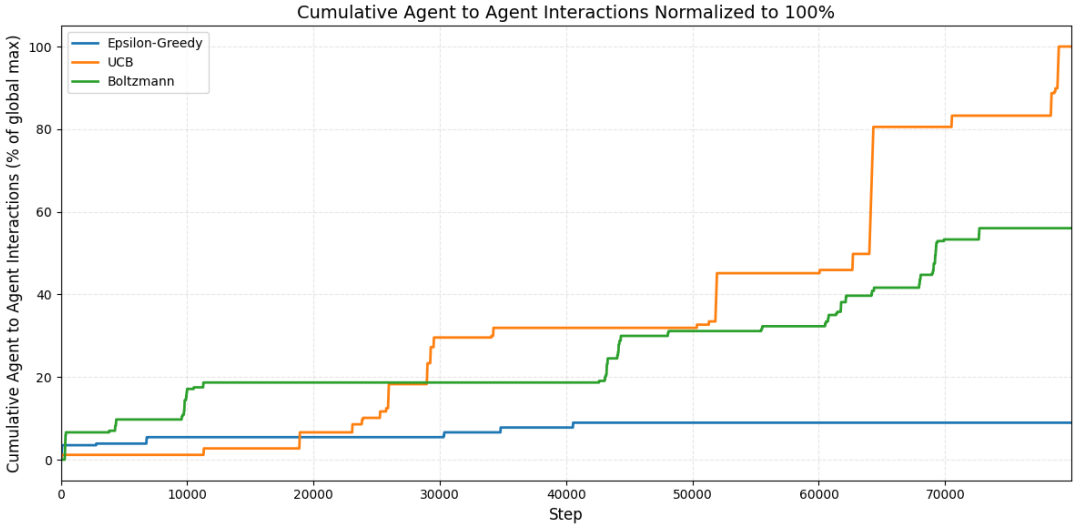


Figure 22: Cumulative Agent to Agent Interactions per policy

4.2 Comparing single to multi-agent

For the best-performing policy, we conducted a direct comparison of the curiosity rewards, as shown in Figure 23. This plot displays the individual curiosity reward of the single-agent setup in comparison to the two curiosity rewards from the multi-agent setup. It is evident that the curiosity rewards in the multi-agent scenario are significantly higher. This is due to the fact that the environment is considerably more dynamic and complex. Since each agent moves independently and can observe the other, predicting the environment becomes much more difficult. As a result, there is constant change, unlike in the single-agent scenario, and this is reflected in the plot.
Moreover, an agent can not only observe the other agent, but also the objects that are being moved by the other agent. This means that an agent can receive a high curiosity reward even when the changes in the environment are caused by the actions of the other agent, rather than its own. In contrast, in the single-agent setup, an object moves only when the agent interacts with it directly. Otherwise, the environment remains static and therefore less stimulating.

TK
LS
AT

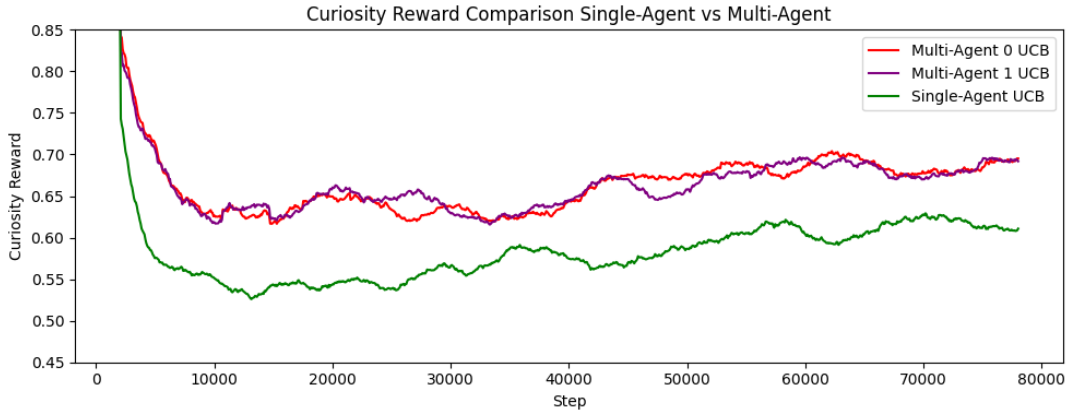


Figure 23: Curiosity Reward comparison Single-Agent vs Multi-Agent

When examining Figures 8 and 17, it becomes apparent that the epsilon-greedy runs are similar to each other.

Specifically, the curiosity reward behaves similarly in both cases, in contrast to the UCB runs, which show a significantly higher and more distinct curiosity reward.

4.2.1 Comparing the Auto-Encoder

To facilitate a direct comparison, the best-performing UCB policy was selected for both the single-agent and multi-agent experiments. A comparison of the Variational Autoencoder performance between these two environments reveals significant differences in their learning dynamics, as illustrated in Figure 24. TK LS AT

The VAE in the multi-agent setting consistently exhibits a higher reconstruction loss throughout the training process. This is an expected outcome, as the model must learn to encode a more complex and varied data stream, processing observations from two distinct agents within each step.

More importantly, a clear difference in convergence behavior is visible. The VAE loss in the single-agent experiment shows a steady and consistent decrease, indicating that the model is effectively converging to a stable representation of its environment. In contrast, while the multi-agent VAE loss also trends downwards, it remains highly volatile and fails to converge to a stable, low error rate within the observed timeframe. This suggests that the increased complexity of the multi-agent observations poses a significant challenge to the VAE's learning stability.

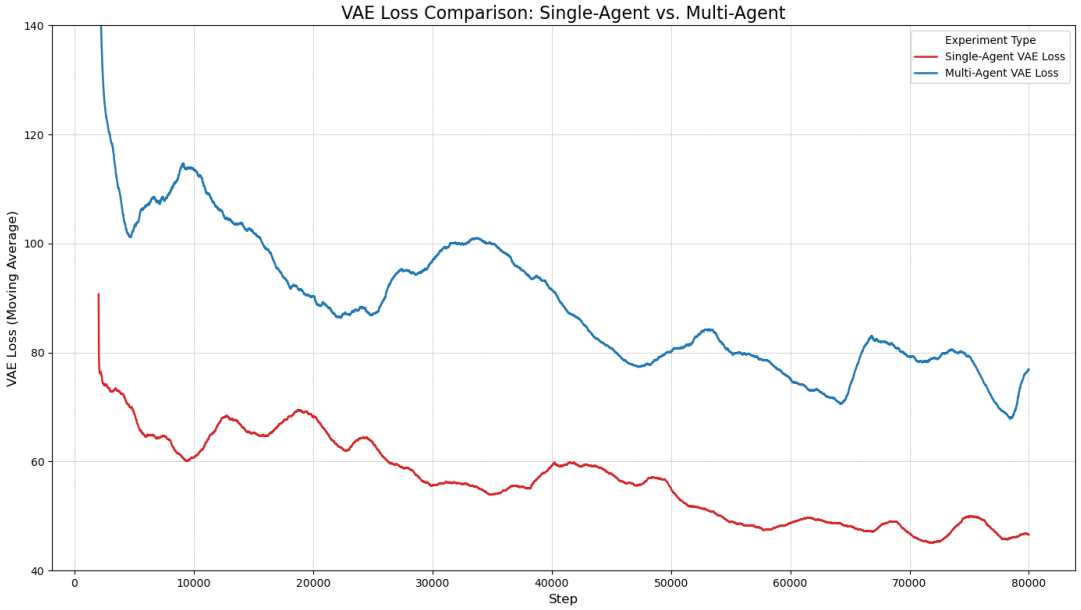


Figure 24: Compare the Autoencoder

5 Discussion

Initially, replicating the original environment from the paper [10] proved extremely challenging, since it was not publicly released and reconstructing it from the available code was difficult. Consequently, we re-implemented the experiments in PyBullet. In our version, the agent initially fixated on the walls, a problem we attempted to resolve by varying wall colors and transparency until we discovered that lighting artifacts in the environment were confusing the autoencoder. Replacing uniform walls with textured patterns eliminated this bias, virtually abolishing wall focus and markedly improving object interactions. During this process, we made a key change to the agent's physical form, switching from a sphere to a cube. This was necessary because the spherical agent's dynamics allowed it to roll indefinitely, making stable control and predictable interactions difficult to learn. The cube, with its stable base, provided more predictable physics for the agent to master. TK LS AT

We also found that the Boltzmann policy would exploit rapid rotations ("trembling") to inflate its intrinsic reward, a behavior that persisted despite appearing optimal to the world model, targeted hyperparameter tuning successfully mitigated this issue for both UCB and ϵ-greedy strategies.

To improve computational efficiency, the world model is updated only every four agent steps, which allows each backpropagation pass to incorporate a broader, more varied batch of experiences. This sparse updating stabilizes training by smoothing non-stationary feedback and substantially reduces runtime, which is why only every fourth step

appears in some plots. Because running 80 000 or more simulation steps on local hardware led to prohibitive runtimes, we migrated to the BW-UniCluster 3.0 and reduced the batch size from 64 to 16. Although the results changed only marginally, the required compute time decreased dramatically, cutting the wall-clock time for 80 000 steps from roughly two days to about five hours. Disabling the GUI further accelerated training. Finally, we record low-resolution egocentric videos for each agent to qualitatively assess performance. These recordings reveal that intrinsic rewards spike whenever objects begin to move (for example, cylinders toppling and rolling between walls), since such newly encountered dynamics generate high prediction errors until the world model has learned them.

At some point during development, we considered implementing the multi-agent setup using parallel programming. While we initially began exploring this approach, we ultimately decided against it due to the inherent risk of race conditions commonly associated with parallel execution.

The core of our architecture is adapted from Ha and Schmidhuber's World Model. We adopted its fundamental design principle: decoupling a large, complex predictive model of the world from a small, simple controller. This separation offers a critical advantage for stability. The world model, consisting of a Vision Model (VAE) and a Memory Model (MDN-RNN), can be trained in a stable, unsupervised manner to learn a compressed representation of the environment's spatial and temporal dynamics. However, a stable world model alone does not solve the exploration problem. In an open-ended environment, especially one with multiple agents, the space of possible interactions is vast. A random or unguided policy would be exceptionally inefficient, rarely stumbling upon the meaningful, cooperative behaviors we wish to foster.

This is where the contribution of Haber et al. becomes essential. We integrate their concept of a "self-model"—an explicit mechanism for the agent to predict its own world model's errors. The policy is then driven by a desire to maximize this predicted error, effectively creating an agent that is intrinsically motivated to seek out situations it does not yet understand.

By combining these two ideas, we create a synergistic system. The Schmidhuber-style world model provides the stable predictive foundation required for learning, while the Haber-style self-model and curiosity-driven policy provide the engine for intelligent and efficient exploration.

A direct comparison of exploration policies revealed that the UCB strategy was significantly more effective at generating interactions than both ϵ-greedy and Boltzmann policies. As shown in Figures 21 and 22, UCB led to the highest cumulative frequency of both agent-object and agent-to-agent interactions, making it the most potent driver for exploration in our setup.

The introduction of a second agent fundamentally changes the learning problem. Our results consistently show that the curiosity reward is significantly higher and the VAE reconstruction loss fails to fully converge in the multi-agent setting (Figures 23 and 24). This indicates that the environment is more dynamic and less predictable, providing a richer, although more challenging, learning signal for the shared world model.

A key finding in our multi-agent experiments is that the curiosity rewards of both agents follow remarkably similar trajectories. This suggests a synchronized learning process mediated by the shared world model. Furthermore, qualitative analysis of the agent videos shows that an agent's curiosity can be triggered simply by observing the novel actions of the other agent, demonstrating a capacity for passive, observational learning.

References

[1] Arthur Aubret, Laetitia Matignon, and Salima Hassas. *A survey on intrinsic motivation in reinforcement learning*. 2019. arXiv: 1908.06976 [cs.LG]. URL: https://arxiv.org/abs/1908.06976.

[2] Marc Bellemare et al. "Unifying count-based exploration and intrinsic motivation". In: *Advances in neural information processing systems* 29 (2016).

[3] Daniel E Berlyne. "Conflict, arousal, and curiosity." In: (1960).

[4] Yuri Burda et al. "Large-scale study of curiosity-driven learning". In: *arXiv preprint arXiv:1808.04355* (2018).

[5] Lucian Busoniu, Robert Babuska, and Bart De Schutter. "Multi-agent reinforcement learning: A survey". In: *2006 9th international conference on control, automation, robotics and vision*. IEEE. 2006, pp. 1–6.

[6] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. "Multi-agent reinforcement learning: An overview". In: *Innovations in multi-agent systems and applications-1* (2010), pp. 183–221.

[7] Lorenzo Canese et al. "Multi-agent reinforcement learning: A review of challenges and applications". In: *Applied Sciences* 11.11 (2021), p. 4948.

[8] Robert L Fantz. "Visual experience in infants: Decreased attention to familiar patterns relative to novel ones". In: *Science* 146.3644 (1964), pp. 668–670.

[9] David Ha and Jürgen Schmidhuber. "World Models". In: *CoRR* abs/1803.10122 (2018). arXiv: 1803.10122. URL: http://arxiv.org/abs/1803.10122.

[10] Nick Haber et al. *Learning to Play with Intrinsically-Motivated Self-Aware Agents*. 2018. arXiv: 1802.07442 [cs.LG]. URL: https://arxiv.org/abs/1802.07442.

[11] Danijar Hafner et al. *Learning Latent Dynamics for Planning from Pixels*. 2019. arXiv: 1811.04551 [cs.LG]. URL: https://arxiv.org/abs/1811.04551.

[12] Danijar Hafner et al. *Mastering Atari with Discrete World Models*. 2022. arXiv: 2010.02193 [cs.LG]. URL: https://arxiv.org/abs/2010.02193.

[13] Danijar Hafner et al. *Mastering Diverse Domains through World Models*. 2024. arXiv: 2301.04104 [cs.AI]. URL: https://arxiv.org/abs/2301.04104.

[14] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2022. arXiv: 1312.6114 [stat.ML]. URL: https://arxiv.org/abs/1312.6114.

[15] Vijay Konda and John Tsitsiklis. "Actor-critic algorithms". In: *Advances in neural information processing systems* 12 (1999).

[16] Mikko Lauri, David Hsu, and Joni Pajarinen. "Partially Observable Markov Decision Processes in Robotics: A Survey". In: *IEEE Transactions on Robotics* 39.1 (Feb. 2023), pp. 21–40. ISSN: 1941-0468. DOI: 10.1109/tro.2022.3200138. URL: http://dx.doi.org/10.1109/TRO.2022.3200138.

[17] Boyi Liu et al. "Neural trust region/proximal policy optimization attains globally optimal policy". In: *Advances in neural information processing systems* 32 (2019).

[18] Fan-Ming Luo et al. *A Survey on Model-based Reinforcement Learning*. 2022. arXiv: 2206.09328 [cs.LG]. URL: https://arxiv.org/abs/2206.09328.

[19] Thomas M. Moerland, Joost Broekens, and Catholijn M. Jonker. "Model-based Reinforcement Learning: A Survey". In: *CoRR* abs/2006.16712 (2020). arXiv: 2006.16712. URL: https://arxiv.org/abs/2006.16712.

[20] Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V Hafner. "Intrinsic motivation systems for autonomous mental development". In: *IEEE transactions on evolutionary computation* 11.2 (2007), pp. 265–286.

[21] Deepak Pathak et al. "Curiosity-driven exploration by self-supervised prediction". In: *International conference on machine learning*. PMLR. 2017, pp. 2778–2787.

[22] 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*. 1991, pp. 222–227.

[23] Jürgen Schmidhuber. "Formal Theory of Creativity, Fun, and Intrinsic Motivation (1990–2010)". In: *IEEE Transactions on Autonomous Mental Development* 2.3 (2010), pp. 230–247. DOI: 10.1109/TAMD.2010.2056368.

[24] Satinder Singh et al. "Intrinsically Motivated Reinforcement Learning: An Evolutionary Perspective". In: *IEEE Transactions on Autonomous Mental Development* 2.2 (2010), pp. 70–82. DOI: 10.1109/TAMD.2010.2051031.

[25] Chenyu Sun, Hangwei Qian, and Chunyan Miao. "From psychological curiosity to artificial curiosity: Curiosity-driven learning in artificial intelligence tasks". In: *arXiv preprint arXiv:2201.08300* (2022).

[26] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*. 2015.

[27] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. "Multi-agent reinforcement learning: A selective overview of theories and algorithms". In: *Handbook of reinforcement learning and control* (2021), pp. 321–384.

[28] Zheng Zhu et al. *Is Sora a World Simulator? A Comprehensive Survey on General World Models and Beyond*. 2024. arXiv: 2405.03520 [cs.CV]. URL: https://arxiv.org/abs/2405.03520.