Compositional Skill Chaining and Policy Blending for Hard Exploration in the BRIO Labyrinth Game

Young-Min Kim [1] **Bo-Yeong Kang** [2]

Abstract

The classic BRIO Labyrinth game presents a hard-exploration problem with sparse and deceptive rewards, requiring precise motor control and long-horizon planning that existing approaches struggle to solve without human guidance. To address this, we propose a two-phase compositional framework. Phase 1 leverages the Go-Explore paradigm to discover a high-performing trajectory. Phase 2 robustifies this trajectory via backward learning and decomposes it into specialized skills using compositional skill chaining, avoiding the instability of a monolithic policy. To ensure kinematic continuity, we introduce a policy-blending mechanism that mitigates control instability during abrupt skill transitions. Experimental results demonstrate our framework effectively solves this challenge, with policy blending actively counteracting physical momentum to achieve a 92.6% success rate, substantially outperforming standard skill chaining.

1. Introduction

Reinforcement Learning (RL) has recently achieved significant progress, particularly in hard exploration domains such as Montezuma's Revenge and Pitfall from the Atari suite (Ecoffet et al., 2021; Salimans & Chen, 2018), Pommerman (Resnick et al., 2018; Gao et al., 2019), and VizDoom (Mikhaylova et al., 2022). These environments typically fall into two categories: sparse reward and deceptive reward problems (Hong et al., 2018).

As illustrated in Fig. 1, in the sparse reward environments (left grid), the agent receives little to no feedback during the entire exploration except when it reaches the goal (green cell). As a result, the agent must execute a long and precise sequence of actions without intermediate guidance. In the absence of feedback, it is extremely challenging for the agent to identify which actions lead to success, resulting in inefficient, slow exploration.

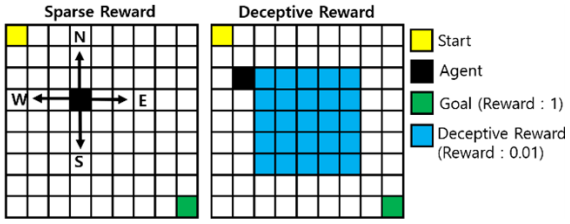


Figure 1. Illustration of sparse and deceptive reward problems in reinforcement learning.

In deceptive reward environments (right grid), the agent receives frequent intermediate rewards (blue cells) that are easy to obtain but do not lead directly to the goal. These intermediate rewards create a local-optimum trap, where agents tend to prefer collecting numerous small deceptive rewards rather than discovering the true goal. Although the agent appears to receive positive feedback, this signal is misleading and can prevent convergence toward the optimal trajectory leading to the final reward.

In this study, we consider the BRIO Labyrinth game, popularized in 1946 by BRIO, a Swedish company renowned for its wooden toys. The game is a physically interactive environment in which the player guides a ball from a start position to a goal by tilting a maze board, while avoiding multiple holes. Even experienced human players often find the game challenging, as it requires fine motor control, long-term planning, and strong spatial reasoning abilities. The real game and its simulated counterpart used in our experiments are shown in Fig. 2 (a) and (b), respectively.

We hypothesize that the BRIO Labyrinth game constitutes an inherently hard-exploration domain. The agent receives a binary reward only upon reaching the goal, resulting in extremely sparse and uninformative feedback. Designing a shaped reward to encourage intermediate progress is also problematic. For instance, using Euclidean distance to the

[1]Department of Robot and Smart System Engineering, Kyungpook National University, Daegu 41566, Republic of Korea [2]Department of Smart Mobility Engineering, Kyungpook National University, Daegu 41566, Republic of Korea. Correspondence to: Bo-Yeong Kang <kby09@knu.ac.kr>.

Accepted to the 2nd Workshop on Compositional Learning at ICML 2026, Seoul, South Korea. Copyright 2026 by the author(s).



(a) The BRIO Labyrinth game (b) The simulation of Labyrinth game

Figure 2. (a) shows the real BRIO Labyrinth game, (b) shows the simulated labyrinth environment used in our experiments.

goal as a reward signal can be misleading, because the maze's structure often requires temporarily moving away from the goal. However, a distance-based reward discourages such behavior, causing the agent to get stuck in dead ends or local optima. This phenomenon highlights both the sparse and deceptive nature of the reward landscape in this environment.

To address the challenges posed by the BRIO Labyrinth game, we adopt a Go-Explore-based reinforcement learning framework that integrates backward learning (Salimans & Chen, 2018) and skill chaining (Konidaris & Barto, 2009). Specifically, we employ Go-Explore (Ecoffet et al., 2021), a method explicitly designed for hard-exploration environments, to discover a successful trajectory from the initial state to the goal. Instead of training directly from the initial state, which would suffer from severe reward sparsity, we leverage backward learning to construct a curriculum that gradually shifts the starting point backward along the discovered trajectory. While this strategy mitigates the sparse reward problem, training a single policy with proximal policy optimization (PPO) (Schulman et al., 2017), a widely used and stable policy gradient method, can become unstable when applied to long-horizon trajectories. To address this limitation, we incorporate skill chaining, which decomposes the full trajectory into manageable segments and learns a set of local policies. To further improve robustness and ensure smooth policy composition, we introduce policy blending to support smooth skill transitions, effectively reducing abrupt switching between adjacent skills across complex maze structures.

2. Related Work

In recent years, classical machine learning and traditional control theory approaches have been widely applied to the BRIO Labyrinth game. For instance, Ofjäll and Felsberg (Öfjäll & Felsberg, 2016) combined deterministic trajectory planning with a locally weighted projection regression (LWPR) model trained via PID controllers. Similarly, Frid et al. (Frid & Nilsson, 2020) employed a gain-scheduled linear-quadratic regulator (LQR) to track human-specified trajectory points interpolated using cubic splines. More recently, Gaber et al. (Gaber et al., 2024) proposed an adaptive nonlinear model predictive control (MPC) framework that utilizes the corners of a predefined orthogonal reference path to manage the maze's complex physical constraints. While demonstrating strong trajectory-tracking capabilities, these approaches fundamentally rely on human-provided reference trajectories or structural priors, and are often evaluated in simplified environments lacking complex hazards like holes.

Several reinforcement learning (RL) frameworks have also been proposed to tackle this environment. Initial attempts (Abdenebaoui et al., 2007; Metzen et al., 2009) established the feasibility of learning-based control using hierarchical Q-learning or SARSA with a cerebellar model articulation controller (CMAC). However, their evaluations were restricted to navigating toward manually defined, short-range waypoints rather than solving the full-scale labyrinth. More recent advancements (Bi et al., 2024; 2025) introduced sample-efficient model-based RL using DreamerV3, augmented with data transformations and selective reconstruction, significantly reducing real-world interaction time. Nevertheless, similar to classical methods, these advanced RL approaches still assume a predefined reference trajectory manually designed on the board, limiting full autonomy.

In summary, although prior research has achieved considerable progress, it remains fundamentally constrained by its reliance on manually defined trajectories. Moreover, several studies circumvent the full complexity of the task by validating their approaches only on partial trajectory segments between manually specified waypoints or on simplified physical environments, such as hole-free mazes. While identifying a feasible path constitutes the central challenge in hard-exploration environments, providing explicit human guidance, such as predefined solution trajectories, substantially reduces task difficulty by effectively transforming a sparse, deceptive reward problem into a path-following task. Moreover, providing manually defined waypoints or designing simpler environments artificially densifies the reward signal. Consequently, these approaches bypass the sparse and deceptive reward problem rather than solving it.

To address these limitations, our work introduces a two-phase framework. First, to address the critical limitations of manual trajectory definition and autonomous exploration in sparse reward environments, we employ the Go-Explore framework. This approach decouples exploration from policy learning, enabling the agent to discover a successful trajectory even in the sparse reward environment of the labyrinth game. In the second phase, to transform the discovered trajectory into a stable and robust control policy, we propose a robustification strategy combining skill chaining

and policy blending. Skill chaining mitigates the instability of a monolithic policy by decomposing the complex trajectory into manageable segments. Furthermore, the policy blending component ensures smooth transitions between skills, which is essential for maintaining stability in high-inertia dynamical systems.

3. A Proposed Go-Explore Framework with Skill Chaining and Policy Blending for Labyrinth Game

We formulate the control problem for the BRIO Labyrinth game as a hard-exploration problem with sparse, deceptive rewards. In this section, we present the methodology developed to address this challenge. As illustrated in Fig. 3, our approach adopts the two-phase Go-Explore framework (Ecoffet et al., 2021; Konidaris & Barto, 2009), incorporates skill chaining and extends it by introducing a policy blending mechanism. This framework is composed of two successive phases: (1) Trajectory exploration and (2) Reinforcement learning with robust skill chaining and policy blending. We first describe the algorithm for Phase 1, which employs a Go-Explore-based exploration strategy to explore the environment and discover a high-performing trajectory. Next, we explain how this trajectory serves as a foundation for Phase 2, in which reinforcement learning and skill chaining are integrated to learn a set of skill-specific policies that robustly follow the identified trajectory. Finally, we propose policy blending for smooth skill transitions, and detail how this method enables the smooth and robust composition of the learned skills to solve the final task.

3.1. Phase 1: Trajectory Exploration

3.1.1. Cell Representations

The objective of Phase 1 is to explore the environment and identify a high-performing trajectory that successfully reaches the final goal state. This trajectory subsequently serves as the backbone for skill acquisition in Phase 2. To accomplish this, inspired by the Go-Explore algorithm, we employ a "cell representation" to simplify the continuous, high-dimensional state space by mapping each state to a compact cell. This approach substantially improves exploration efficiency.

Specifically, at each timestep t, the environment state is rendered as an image frame $o_t \in \mathbb{R}^{H\times W\times 3}$. Using the raw image frame, which has high resolution (e.g., 100×100 pixels) and RGB channels, is inefficient for exploration algorithms like Go-Explore. In continuous control environment like the BRIO Labyrinth game, even minute agent actions involve slight pixel-level variations that may be interpreted as entirely new states. This phenomenon leads to a combinatorial explosion of the state space, exacerbating the exploration challenge and undermining Go-Explore's core mechanism of recognizing previously visited states. To address this limitation, we apply the following preprocessing steps to derive a compact cell representation c_t from the raw observation o_t:

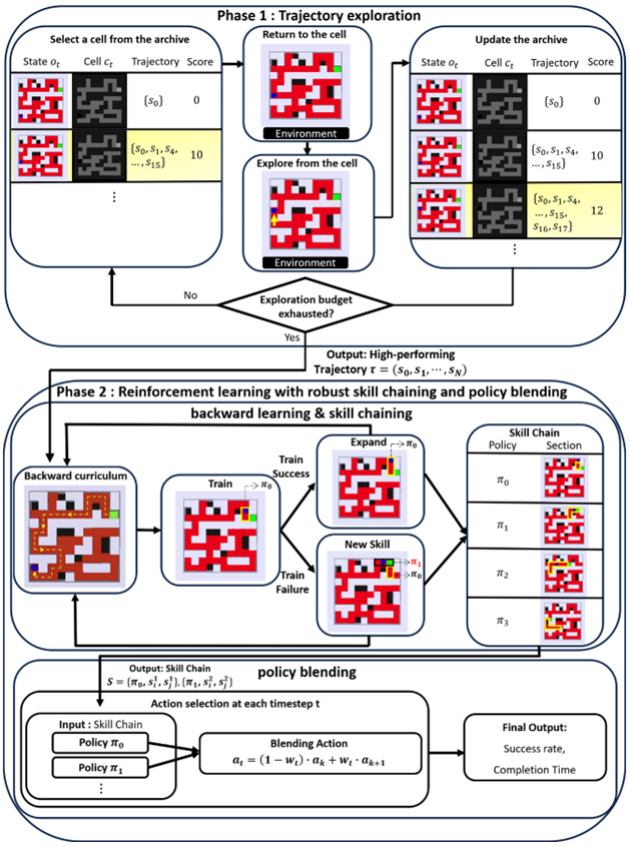


Figure 3. An overview of the proposed two-phase Go-Explore framework for labyrinth game.

1. **Grayscale conversion**: The RGB observation o_t is converted into a grayscale image $\tilde{o}_t \in \mathbb{R}^{100\times 100}$ by averaging across the color channels.

2. **Downsampling**: The grayscale image $\tilde{o}_t$ is resized to a low-resolution representation $\tilde{o}_t^{small} \in \mathbb{R}^{M\times M}$, where $M \times M$ corresponds to the grid resolution of the maze.

3. **Pixel intensity rescaling**: Each pixel value in $\tilde{o}_t^{small}$, which is originally in the range $[0, 255]$, is linearly rescaled into an integer value within the range $[0, 15]$, forming an integer matrix $c_t \in \{0, 1, \ldots, 15\}^{M\times M}$.

The resulting low-resolution integer matrix, c_t, provides a compressed representation of the original observation while substantially reducing the size of the state space. This abstraction is particularly beneficial for Go-Explore, as it enables efficient exploration with a discrete state space and

facilitates reliable recognition of previously visited states for archive management. Consequently, this representation supports effective trajectory discovery in the BRIO labyrinth environment despite its continuous action space.

3.1.2. Explore Until Solved

In Phase 1, the agent explores the environment to discover a high-performing trajectory that reaches the goal state. This section outlines the overall exploration procedure. Throughout the process, an archive is maintained to record all visited states and the trajectories leading to them. For each entry, the archive stores: (1) the original observation o_t, (2) the corresponding cell c_t, (3) the full trajectory from the initial state to c_t, and (4) the trajectory length. This length serves as a scoring metric for archive updates, such that an entry associated with a given cell is replaced only if a newly discovered trajectory is shorter than the existing one.

1. **Select a cell from the archive**: A cell c_t is sampled from the archive based on a sampling strategy. In this study, we employ uniform random sampling without heuristic bias.
2. **Return to the cell**: The agent deterministically returns to the selected cell by resetting the environment to the stored state corresponding to c_t, without performing any additional exploration during this process.
3. **Explore from the cell**: Starting from c_t, the agent performs stochastic exploration for a fixed number of steps, aiming to discover new cells or improved trajectories to existing ones.
4. **Update the archive**: If a new cell is discovered or a shorter trajectory to an existing cell is identified, the archive is updated accordingly.

The exploration phase proceeds iteratively for a predefined total number of steps, progressively expanding the archive with higher-performing trajectories. Once this step limit is reached, the algorithm outputs the best-performing (shortest) trajectory to the goal state contained in the final archive. This trajectory serves as the foundation for subsequent skill acquisition and robust policy learning in Phase 2.

3.2. Phase 2: Reinforcement Learning with Robust Skill Chaining and Policy Blending

3.2.1. Problem Formulation

The output of Phase 1 is a high-performing trajectory $\tau = (s_0, s_1, \ldots, s_N)$, where each s_i denotes the state corresponding to the cell visited along the trajectory from the start state s_0 to the goal state s_N. The primary objective of Phase 2 is to train a policy to robustly follow this discovered trajectory. To this end, the sequence of states is used as a curriculum for reinforcement learning, where the associated physical (x, y) coordinates in the BRIO labyrinth simulation serve as initialization points during training.

Within this simulation framework, the control problem is formulated as a continuous interaction process governed by physical dynamics. At each time step t, the agent receives a visual observation o_t, represented as a top-down RGB image capturing the current maze configuration and the ball's position. Based on this observation, the policy π outputs a continuous action a_t, corresponding to the target roll and pitch angles of the board. These control signals physically tilt the labyrinth board, inducing forces on the ball governed by ball dynamics such as gravity, inertia, and friction. This interaction results in a state transition to s_{t+1}, providing the agent with a new observation and a reward signal R_t. The objective of the policy is to manipulate the board tilts to accurately guide the ball along the trajectory discovered in Phase 1.

3.2.2. Robustification via Backward Learning and Skill Chaining

A naïve reinforcement learning approach trains a single monolithic policy $\pi(a|s)$ to reach the final state s_N starting from the initial state s_0. However, this formulation suffers from the hard-exploration problem: for an agent initialized at s_0, the goal state s_N is distant, resulting in extremely sparse rewards and prohibitively inefficient learning.

To address this challenge, we adopt backward learning. Instead of beginning at the difficult initial state s_0, the training curriculum is constructed in reverse, starting from the end of the trajectory and progressing backward. The agent first learns the simplest sub-task: reaching the final goal state s_N from the nearest preceding waypoint s_{N-1}. Once this sub-policy is successfully learned, the curriculum expands by shifting the starting state further backward to s_{N-2}, requiring the agent to reach the same goal state s_N, from a more distant state. This procedure is repeated by progressively increasing the initial distance until the agent can reliably reach s_N from the original starting state s_0. This strategy ensures that the agent always begins training episodes from states where the goal is locally accessible, thereby providing a dense and informative learning signal. Consequently, a single, difficult, sparse-reward problem is transformed into a sequence of simpler dense-reward problems, fundamentally improving learnability in hard exploration domains.

Although this backward curriculum mitigates the sparse reward issue, learning a single policy over the resulting long-horizon trajectory still imposes limitations due to representation complexity. In particular, encoding the full control dynamics of an extended trajectory within a single global function approximator places an excessive burden on the neural network. This often leads to suboptimal performance,

Algorithm 1 Skill Chaining via Backward Learning

1: **Input:** A trajectory $\mathcal{T} = (s_0, s_1, \ldots, s_N)$
2: **Parameters:** Threshold $\theta_{success}$, Attempts $N_{attempts}$
3: **Initialize:**
4: Skill set $\mathcal{S} \leftarrow \emptyset$
5: Skill index $k \leftarrow 0$
6: Current policy $\pi_k \leftarrow$ Init()
7: Target state $s_{target} \leftarrow s_N$
8: Current state index $i \leftarrow N - 1$
9: **while** $i \geq 0$ **do**
10: $attempts \leftarrow 0$
11: $success_rate \leftarrow$ Evaluate(π_k, s_i, s_{target})
12: **while** $success_rate < \theta_{success}$ **and** $attempts < N_{attempts}$ **do**
13: Train π_k on subtask $s_i \rightarrow s_{target}$
14: $success_rate \leftarrow$ Evaluate(π_k, s_i, s_{target})
15: **if** $success_rate \geq \theta_{success}$ **then**
16: $i \leftarrow i - 1$
17: **break**
18: **end if**
19: $attempts \leftarrow attempts + 1$
20: **end while**
21: **if** $attempts \geq N_{attempts}$ **then**
22: $\mathcal{S} \leftarrow \mathcal{S} \cup \{\pi_k\}$
23: $s_{target} \leftarrow s_{i+1}$
24: $k \leftarrow k + 1$
25: $\pi_k \leftarrow$ Init()
26: **end if**
27: **end while**
28: **return** Skill set $\mathcal{S}$

as the model struggles to capture the distinct dynamics of different trajectory segments simultaneously.

To address this limitation, we employ a skill chaining methodology. The central idea is to decompose the full trajectory into a sequence of manageable segments, each with its own specialized policy, called a skill. This decomposition reduces the burden on the function approximator by allowing each policy to focus on a localized and structurally simpler control problem.

The procedure for skill chaining is summarized in Algorithm 1. The algorithm begins by initializing an empty skill set $\mathcal{S}$ and the first skill policy π_0 (lines 4-6). The initial target state (s_{target}) is set to the final goal s_N, and the process starts backward from the penultimate state s_{N-1} (lines 7-8).

Inside the main while loop, the current skill policy π_k attempts to master the segment from the current state s_i to the target state s_{target}. First, the policy's performance is evaluated immediately before any training is conducted (line 11). If the success rate falls below a predefined threshold $\theta_{success}$, an inner training loop (lines 12-20) is activated to train the skill policy π_k. The train procedure (line 13) employs reinforcement learning to solve this sub-task. We formalize the transition $s_i \rightarrow s_{target}$ as a Markov decision process (MDP), defined by the tuple $\mathcal{M}_k = (S, A, R_k, P, \gamma)$. In this formulation, the state space S consists of high-dimensional visual observations, specifically top-down images of the labyrinth, requiring the agent to infer the ball's position and velocity directly from pixel data. The action space A is continuous and represents the inclination of the board along two axes. The agent controls the ball by tilting the board, thereby manipulating gravity to navigate the maze. The agent receives a terminal reward of $+1$ upon successfully reaching the target state. Additionally, to encourage continuous progress towards the goal at each time step t, the agent is rewarded based on the reduction in Euclidean distance between its previous and current states:

$$R_k = \begin{cases} +1, & \text{if } s_{t+1} = s_{target} \\ d(s_t, s_{target}) - d(s_{t+1}, s_{target}), & \text{otherwise} \end{cases}$$

This reward function provides an immediate positive signal when the agent moves closer to the target and a negative signal when it moves farther away. To train this sub-task MDP, we adopt a policy-based reinforcement learning approach. The BRIO labyrinth game features a continuous action space in which the board must be tilted to fine-grained angles. These characteristics make value-based methods, such as Q-learning or DQN, typically requiring discrete action approximations, difficult to apply directly. Therefore, we employ proximal policy optimization (PPO), a robust actor-critic method widely used for continuous control tasks. It is worth noting that while using Euclidean distance as a global reward signal can be deceptive in complex mazes, it functions effectively within our sub-task MDPs. Because the dynamic skill chaining decomposes the full long-horizon trajectory into local, structurally simple segments, the target waypoint s_{target} is always in close proximity to the initialization state s_i. Within these localized sub-regions, the directional guidance provided by the Euclidean distance aligns consistently with the optimal path without introducing structural deception, thereby facilitating stable and efficient policy gradients.

After this inner training loop terminates, the algorithm evaluates the outcome. If training is successful, i.e., if the success rate is greater than or equal to $\theta_{success}$, it indicates that the current skill π_k has successfully learned the extended segment. The algorithm expands the current skill by decrementing i (line 16) and proceeds to the next backward step s_{i-1}, using the same policy π_k.

If training fails, i.e., if the number of attempts reaches or exceeds $N_{attempts}$, it indicates that the current skill π_k has reached its limit in mastering the sub-task. At this point, dynamic partitioning is triggered. The algorithm adds π_k to the skill set $\mathcal{S}$ and sets s_{i+1} as the new target, s_{target}.

Finally, a new skill π_{k+1} is initialized with fresh parameters (line 21-25).

Initializing a new skill π_{k+1} ensures that training does not interfere with the parameters of the previously learned skill π_k. This strategy mitigates the representational burden on the neural network. By decomposing the long-horizon task, it avoids the difficulty of encoding the entire complex trajectory with a single global function approximator, allowing each skill to focus on learning a simpler, locally constrained control policy. Detailed information regarding the neural network architecture and implementation parameters is provided in Appendix A.

3.2.3. Policy Blending for Smooth Policy Transition

While the skill set generated by Algorithm 1 effectively partitions the trajectory and improves training stability, it introduces a potential issue during execution. At the boundary between two consecutive skills, the agent must abruptly switch control from π_k to π_{k+1}. This hard switching can lead to control instability in environments such as the BRIO labyrinth game, where physical dynamics, including inertia and momentum, play a substantial role. For example, if one skill applies a strong leftward tilt to move the ball, and the subsequent skill requires an abrupt upward tilt to navigate a corner, the system does not account for the ball's momentum under the new policy. This abrupt transition may cause the ball to deviate from the intended trajectory, resulting in overshooting or sliding at corners and thus destabilizing control.

To address this issue and ensure kinematic continuity between skills, we propose policy blending for smooth skill transitions. The core idea is to enable a gradual transition by smoothly interpolating between the policies of two adjacent skills, rather than transferring control instantaneously at the skill boundary. This produces a smoother control signal, allowing the ball to follow a more continuous and stable trajectory.

Specifically, when the agent operates within the segment assigned to skill π_k, the final action a_t is computed as a weighted combination of the action proposed by the current skill a_k and that proposed by the subsequent skill a_{k+1}:

$$a_t = (1 - w_t)a_k + w_t a_{k+1} \tag{1}$$

Here, the blending weight w_t is defined based on the agent's relative position within the segment assigned to the current skill. Let L_k denote the length of this segment, and let x_{local} represent the local index of the point on the reference trajectory closest to the agent's current position. The blending weight w_t is then computed as:

$$w_t = \left(\frac{x_{local} + 1}{L_k} \right)^k \tag{2}$$

In this formulation, the exponent k acts as a shaping parameter that governs the transition's nonlinearity. Larger values of k allow the agent to retain the influence of the current policy for a longer portion of the segment, followed by a rapid yet smooth transition as the agent approaches the segment boundary. Based on empirical tuning in the BRIO Labyrinth environment, we set $k = 4$ to balance trajectory tracking performance and transition stability.

Under this scheme, when the agent is near the beginning of the segment associated with the current skill, w_t is close to 0, and the current policy dominates the action. As the agent progresses towards the end of the segment, w_t approaches 1, and the influence of the subsequent policy gradually increases. This progressive blending ensures that, by the time control formally transfers to the next skill, both policies are already closely aligned, enabling a smooth and stable transition in physical execution. Experimental results confirm that this conceptually simple blending strategy substantially improves control stability and task success rates in dynamic environments such as the BRIO Labyrinth game, where physical effects like inertia play a dominant role.

4. Experiments

In this section, we present experiments conducted on four distinct BRIO labyrinth maps with varying scales and levels of complexity to validate the effectiveness of the proposed approach. These maps are specifically designed to evaluate the proposed two-phase framework, assessing capabilities that range from long-horizon trajectory planning to robust trajectory following under complex dynamic constraints. We first analyze the performance of Go-Explore combined with skill chaining to demonstrate its ability to solve the hard-exploration problem in the labyrinth environment. Furthermore, we evaluate Go-Explore combined with robust skill chaining and policy blending to examine how policy blending enhances control stability and incrementally improves the overall success rate.

4.1. Experimental Setup

We conducted all experiments using the CoppeliaSim simulator, with the Bullet 2.83 physics engine employed to model the physical BRIO Labyrinth game. To ensure a consistent computational environment, all experiments were performed on a workstation equipped with an Intel i5-4440 CPU and an NVIDIA GeForce GT 720 GPU. The environment provides a 100×100 RGB observation rendered from a top-down perspective. This visual input, together with con-

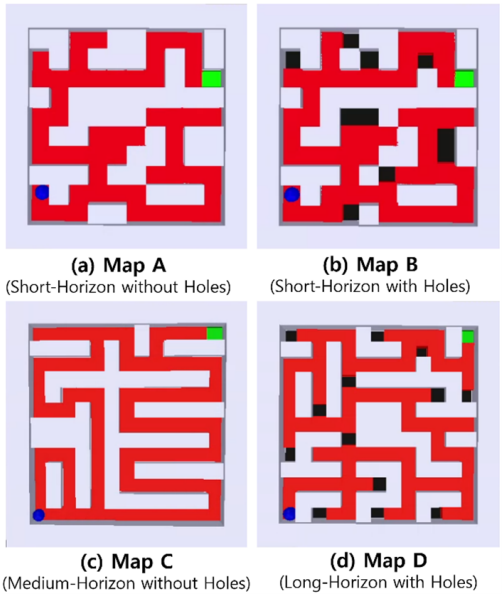


Figure 4. The four simulated environments used in the experiments.

tinuous control actions for board tilting, defines the agent's interaction within the state-action space. The primary challenge lies in precisely managing the ball's momentum to navigate narrow corridors while avoiding hazards such as holes.

To systematically isolate and evaluate the effects of environmental hazards and trajectory length, we constructed four simulated maps (Map A through Map D) with increasing levels of complexity, as shown in Fig. 4. These environments range from short-horizon layouts with solid walls to highly complex, long-horizon mazes featuring multiple paths and hazard holes. A detailed structural description of each map is provided in Appendix B.

Because the proposed framework comprises two phases, we evaluate each phase using distinct metrics. For trajectory exploration (Phase 1), we measure the time required to discover a high-performing trajectory and the archive size, defined as the number of stored cells. For reinforcement learning with robust skill chaining and policy blending (Phase 2), each model is trained over five independent runs with different random seeds. The trained policies are then evaluated over 100 test episodes per run. The primary metric for this phase is the success rate, defined as the percentage of episodes in which the agent reaches the goal without falling into a hazard or exceeding the maximum step limit.

4.2. Results of Trajectory Exploration

In the first phase, we evaluate the Go-Explore algorithm's efficiency at autonomously discovering a high-performing trajectory. Addressing the hard-exploration challenge, namely, navigating complex physical hazards without relying on predefined routes, constitutes a primary contribution of this study. To the best of our knowledge, this study represents the first successful application of the Go-Explore algorithm to the physically constrained BRIO Labyrinth game.

Table 1. Average discovery time and archive size measured during the trajectory exploration (Phase 1)

Environment	Path Length	Avg. Discovery Time (min)	Avg. Archive Size (cells)
Map A	28	25.7	52.0
Map C	38	87.7	77.4
Map D	65	121.0	85.0

To assess the exploration process, we conducted five independent trials, each initialized with a different random seed. Because Map B is structurally identical to Map A except for the presence of holes, it is excluded from this phase. Instead, to examine how the computational cost scales with optimal path length, map size, and archive size, we compare Maps A, C, and D.

As summarized in Table 1, for Map A, which has an optimal path length of 28, the agent discovered a high-performing trajectory in an average of 25.7 minutes, with an archive size of 52.0 cells. Map C exhibits an intermediate optimal path length of 38 and a bifurcated structure, resulting in an average discovery time of 87.7 minutes and an archive size of 77.4 cells. For Map D, which represents the most challenging long-horizon environment with an optimal path length of 65, the agent achieved an average discovery time of 121.0 minutes and an archive size of 85.0 cells. These results demonstrate that the proposed framework can autonomously explore and discover viable trajectories from scratch, effectively handling both single-path and multiple-path structures without relying on human demonstrations or predefined routes.

A key observation from these results is the relationship between the final archive size and the physical structure of the environments. The recorded archive sizes for Map A (52.0 cells) and Map D (85.0 cells) closely approximate the total number of traversable tiles available on each map, which are approximately 55 and 95, respectively. These findings indicate that, unless the agent identifies the optimal path early in the exploration process, the algorithm can exhaustively explore nearly the entire reachable state space. Although such an exhaustive exploration may appear inefficient in conventional settings, it is particularly meaningful in hard-exploration domains, where exploration itself is inherently difficult. Given that the BRIO Labyrinth game constitutes such an environment, these results demonstrate that the Go-Explore approach effectively addresses this challenge, enabling the agent to discover high-performing trajectories.

Table 2. Comparison of success rates between skill chaining and robust skill chaining with policy blending (Phase 2)

Method	Map A	Map B	Map C	Map D
Go-Explore + Skill Chaining	88.4%	67.6%	83.0%	45.6%
Go-Explore + Robust Skill Chaining	**92.6%**	**75.4%**	**92.2%**	**64.6%**

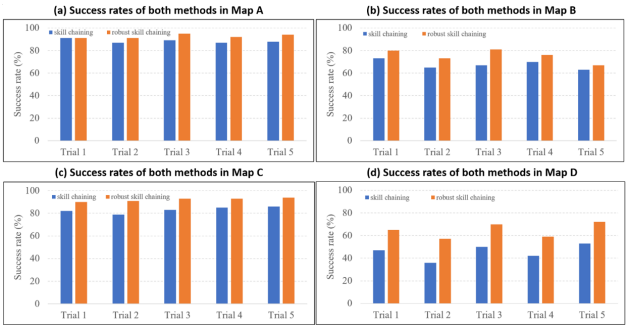


Figure 5. Test success rates of both methods across five trials in the four simulated environments.

4.3. Results of Reinforcement Learning with Robust Skill Chaining and Policy Blending

In the second phase, we evaluate the control stability and task performance of the trained policies. The primary objective of this evaluation is to demonstrate that the baseline framework, Go-Explore combined with skill chaining, successfully solves the hard-exploration BRIO Labyrinth game and to validate that integrating robust skill chaining with policy blending further improves performance. Table 2 summarizes the average success rates of the two methods across four different environments. Fig. 5 further illustrates the distribution of success rates over five independent trials per map, where each trial corresponds to a model trained with a different random seed, thereby providing insight into performance variability. A detailed analysis of the computational cost and training times across these environments is provided in Appendix C.

In Map A, the simple layout and the absence of holes allow both methods to achieve high success rates of 88.4% and 92.6%, respectively. Although Map A features a long straight section that allows the agent to build momentum, it terminates safely at a wall. Consequently, failure modes associated with uncontrolled momentum, such as deviating from the path or falling into a hole, rarely occur. Because momentum control is less critical in this environment, the performance gap between the two methods remains small at 4.2 percentage points. Map B shares the same layout as Map A but includes holes, which increase the risk of failure due to accumulated momentum. As a result, the success rates of both methods decrease to 67.6% and 75.4% (+7.8 percentage points), respectively.

In Map C, there are no holes, and the trajectory is designed to be easily followed, thereby minimizing the likelihood of the agent deviating from the optimal path. As a result, both methods achieve high success rates of 83.0% and 92.2%, respectively. Notably, the performance gap increases to +9.2 percentage points in this environment. This improvement can be attributed to the presence of an extended straight section near a two-way junction, where the agent accumulates significant momentum and inertia. While the baseline skill chaining method exhibits control instability in such high-inertia regions, the robust skill chaining approach effectively mitigates this issue through smoother skill transitions, leading to a substantial performance improvement.

The performance gap becomes most pronounced in Map D, which includes both a long trajectory and multiple hazards. In this environment, the baseline skill chaining method achieves a success rate of 45.6%. In contrast, robust skill chaining combined with policy blending achieves a success rate of 64.6%, an improvement of 19.0 percentage points. This substantial gain is primarily due to the structure of Map D, which contains more segments where the ball accumulates marked momentum and acceleration compared to the baseline maps. Under these conditions, the smooth skill transitions enabled by policy blending effectively alleviate control instability induced by high inertia.

In summary, these experimental results clearly validate the effectiveness of the proposed framework. Unlike prior studies that rely on predefined trajectories, our approach autonomously discovers and traverses complete paths in a hard-exploration setting. Furthermore, the results reveal a clear trend: as environmental complexity increases in terms of obstacles and trajectory length, overall success rates naturally decrease. However, it is precisely under these highly complex and hazardous conditions that policy blending becomes most critical. Compared to relying solely on Go-Explore with basic skill chaining, integrating robust skill chaining with policy blending effectively mitigates the adverse effects of physical inertia, ultimately improving control stability and substantially increasing task success rates across all evaluated environments.

5. Conclusion

In this study, we proposed a two-phase framework to address the dual challenges of high-precision motor control and long-horizon planning in the BRIO Labyrinth game. While previous approaches have often relied on manual human intervention to circumvent the hard-exploration nature of this task, our framework leverages the Go-Explore paradigm to discover high-performing trajectories autonomously.

The experimental results validate the effectiveness of each proposed framework. In Phase 1 (Trajectory exploration), we demonstrate that the Go-Explore paradigm can effi-

ciently identify high-performing trajectories in environments with sparse, deceptive rewards, where standard reinforcement learning methods typically fail. Furthermore, we show that integrating robust skill chaining with policy blending effectively mitigates control instabilities induced by physical inertia, thereby improving task success rates in hazardous environments. These results confirm that structured exploration is essential for solving complex physical puzzle tasks such as the labyrinth game.

In conclusion, this study establishes that integrating autonomous trajectory exploration with stable skill-based reinforcement learning is an effective strategy for physics-driven control problems. Future work will focus on improving training efficiency and investigating framework's generalization to broader dynamic physical systems.

References

Abdenebaoui, L. et al. A connectionist architecture for learning to play a simulated brio labyrinth game. In *Annual Conference on Artificial Intelligence*, 2007.

Bi, T. et al. Sample-efficient learning to solve a real-world labyrinth game using data-augmented model-based reinforcement learning. In *International Conference on Robotics and Automation (ICRA)*, 2024.

Bi, T. et al. Mastering the labyrinth game: Efficient multimodal reinforcement learning with selective reconstruction. In *International Conference on Intelligent Robots and Systems (IROS)*, 2025.

Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K. O., and Clune, J. First return, then explore. *Nature*, 590(7847): 580–586, 2021.

Frid, E. and Nilsson, F. Path following using gain scheduled lqr control with applications to a labyrinth game. 2020.

Gaber, J., Bi, T., and D'Andrea, R. Adaptive nonlinear model predictive control for a real-world labyrinth game. In *63rd IEEE Conference on Decision and Control (CDC)*, 2024.

Gao, C., Pablo, H.-L., Kartal, B., and Taylor, M. E. On hard exploration for reinforcement learning: A case study in pommerman. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 15, pp. 24–30, 2019.

Hong, Z.-W., Shann, T.-Y., Su, S.-Y., Chang, Y.-H., Fu, T.-J., and Lee, C.-Y. Diversity-driven exploration strategy for deep reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Konidaris, G. and Barto, A. Skill discovery in continuous reinforcement learning domains using skill chaining. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2009.

Metzen, J. H., Kirchner, E., Abdenebaoui, L., and Kirchner, F. The brio labyrinth game-a testbed for reinforcement learning and for studies on sensorimotor learning. In *Multidisciplinary Symposium on Reinforcement Learning*, 2009.

Mikhaylova, E. et al. Curiosity-driven exploration in vizdoom. In *20th Jubilee International Symposium on Intelligent Systems and Informatics (SISY)*, 2022.

Öfjäll, K. and Felsberg, M. Combining vision, machine learning and automatic control to play the labyrinth game. In *Swedish Symposium on Image Analysis*, 2016.

Resnick, C., Raileanu, R., Kapoor, S., Peysakhovich, A., Cho, K., and Bruna, J. Backplay: "man muss immer umkehren". *arXiv preprint arXiv:1807.06919*, 2018.

Salimans, T. and Chen, R. Learning montezuma's revenge from a single demonstration. *arXiv preprint arXiv:1812.03381*, 2018.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

A. Network Architecture and Implementation Details

As illustrated in Fig. 6, each skill policy π_k is implemented as a deep neural network using an actor-critic architecture that takes the raw observation image o_t as input. The network comprises a shared feature extractor followed by two separate heads: the actor (policy) and the critic (value function).

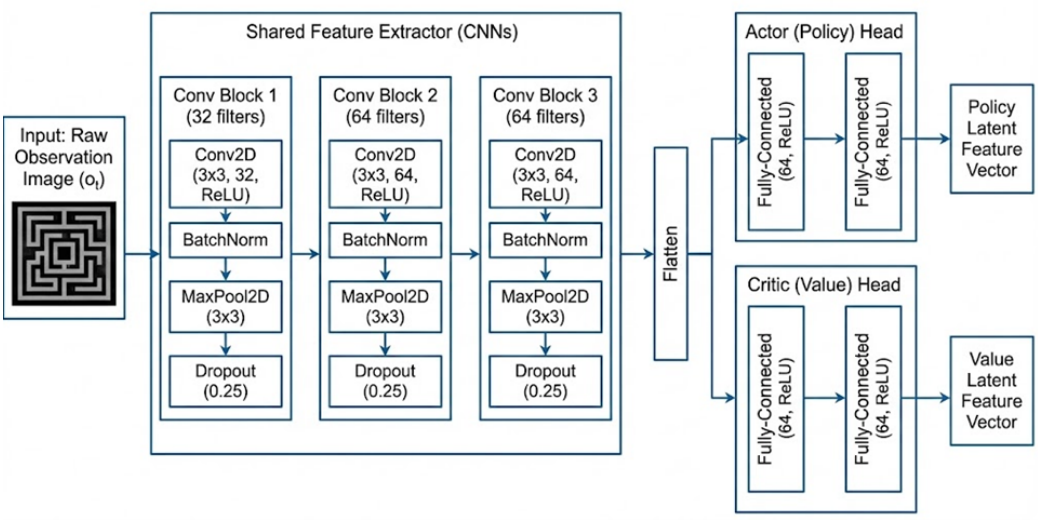


Figure 6. The detailed architecture of the actor-critic network used for skill policies.

The shared feature extractor comprises three sequential convolutional blocks. The first block applies a 2D convolution with 32 filters of size 3×3, followed by a ReLU activation, batch normalization, a 3×3 max pooling layer, and a dropout layer with a rate of 0.25. The subsequent two blocks follow the same structure but increase the number of filters to 64, thereby enhancing the model's representational capacity. This deep convolutional architecture is designed to capture hierarchical spatial features from the input image, including the relative positions of the ball, walls, and holes.

After the convolutional blocks, the resulting feature maps are flattened into a one-dimensional vector, which serves as input to two parallel processing streams:

- **Actor (Policy) head:** The flattened vector is passed through two fully connected layers, each with 64 units and ReLU activations, and is then projected onto a linear output layer that predicts the parameters of a continuous action distribution.
- **Critic (Value) head:** In parallel, the same flattened vector is processed by a separate set of two fully connected layers (64 units, ReLU activations). This stream terminates in a single linear unit that outputs the estimated scalar state-value $V(s_t)$.

B. Experimental Environment Details

To systematically isolate and evaluate the effects of environmental hazards and trajectory length, we constructed four distinct maps with increasing levels of complexity:

- **Map A (Short-Horizon without Holes):** A 12×12 grid environment with an optimal path length of 28, consisting of solid walls and no holes. In this setting, precise momentum control is less critical, making it suitable for evaluating baseline path-following performance.
- **Map B (Short-Horizon with Holes):** A 12×12 grid environment identical in scale to Map A, with an optimal path length of 28, but augmented with holes and sharp curves. The inclusion of holes explicitly tests the agent's ability to maintain precise momentum control. In this context, improper momentum management during skill transitions may cause the agent to fall into a hole or deviate from the intended path.

- **Map C (Medium-Horizon without Holes):** A 15×15 grid environment with an intermediate optimal path length of 38. Unlike the other maps, which provide only a single valid route to the goal, Map C includes two distinct paths: an optimal shorter route and a sub-optimal longer route. This multiple-path structure is specifically designed to evaluate the algorithm's ability to avoid suboptimal trajectories and identify the highest-performing path.
- **Map D (Long-Horizon with Holes):** A larger 15×15 grid environment with an extended optimal path length of 65. Building upon the previous environments, this map combines environmental hazards with a substantially longer trajectory to assess the scalability of the proposed framework. It evaluates whether the agent can reliably chain skills over long horizons without accumulating control errors.

C. Computational Cost Analysis

We also evaluated the computational cost by measuring the average wall-clock time required during the policy-learning phase. As summarized in Table 3, the training time scales directly with environmental complexity.

Map A, the simplest environment without holes, requires approximately 7 hours of training. In Map B, despite having the same grid size and trajectory length as Map A, the average training time nearly doubles to 13 hours. This increase is primarily attributed to the introduction of holes, which require additional training interactions for the agent to learn precise momentum control.

The computational cost further increases in Map C, reaching approximately 24 hours. The main contributing factor is the expansion to a 15×15 grid and an increased optimal path length of 38, both of which require additional training time for successful trajectory acquisition. Finally, Map D demands the highest computational cost, averaging 50 hours per trial. This increase reflects the compounded difficulty of the environment, which combines a larger 15×15 grid, multiple holes, and a considerably longer path length of 65. The interaction of these factors makes maintaining control stability more challenging, thereby significantly increasing the required training time.

Table 3. Average training time (wall-clock time) required during the policy-learning phase across the four environments.

Environment	Size	Path Length	Holes	Avg. Training Time
Map A	12×12	28	No	7 hours
Map B	12×12	28	Yes	13 hours
Map C	15×15	38	No	24 hours
Map D	15×15	65	Yes	50 hours