Decision Step Frequency Impact in Continuous-Time Goal-Conditioned RL within Video Games

Anonymous authors
Paper under double-blind review

Abstract

The decision step frequency, i.e., how often an agent changes its action in a continuous-time environment, is an important design choice when applying reinforcement learning in video games. We study its role in the goal-conditioned setting, where exploration difficulty and sparse success signals make this hyperparameter consequential. In this paper, we present empirical evaluations across four environments: three research-oriented open-source environments inspired by video games, and a complex fully fledged commercial video game. We show that both excessively high and excessively low decision-step frequencies severely degrade performance, inducing sharp drops in success rate and trajectory quality as environment task complexity increases. We also provide a theoretical analysis; and practical guidance for selecting effective frequency ranges via Pareto fronts between convergence speed and trajectory optimality.

1 Introduction

Many real-world RL applications require an agent to operate within complex real-time environments, such as simulation platforms, robotics systems, or game virtual worlds. The latter setting reflects common RL deployment scenarios such as improved quality in non-player characters, player behavior modeling, or automated bug testing (Chambon & Grelier, 2025; Sestini et al., 2024).

These environments, and specifically non turn-based games, evolve in continuous time; however, the agent typically selects actions at discrete decision intervals despite the underlying dynamics being continuous. This mismatch between continuous-world dynamics and discrete decision-making introduces a subtle but fundamental design choice: the decision step frequency, i.e., how often the agent updates its action.

We are interested in the complex goal-conditioned reinforcement learning (GCRL) setting, where training relies on sparse rewards. Unlike dense-reward settings where the agent receive a continuous gradient signal from the first step, setting this frequency too high may prevent the learning process from even starting in GCRL. Conversely, setting it too low may degrade performance by causing the agent to under-react to environmental dynamics, but can lower computational cost (Reda et al., 2020; Srinivas et al., 2017).

On the one hand, related notions such as frame skipping have been studied (Kalyanakrishnan et al., 2021), but they do not fully capture the challenges faced by reinforcement learning practitioners that arise when discrete RL algorithms are embedded in continuous worlds. In real-time control settings, the decision step frequency directly determines the temporal resolution at which the agent can sense and react to modifications in the environment. It therefore alters the interaction dynamics between the agent and environment, rather than merely acting as a conventional hyperparameter.

On the other hand, advanced methods like FiGaR (Sharma et al., 2017), TACOS Treven et al. (2024), or TempoRL Biedenkapp et al. (2021) attempt to learn adaptive action repetitions. These introduce significant architectural and optimization complexity. Therefore, RL practitioners still typically

choose this parameter manually. For instance, Hildisch et al. (2025) train a driving policy at 10 Hz but deploys it at 15 Hz during inference, without formal justification. Specifically in the video game context, Gordillo et al. (2021) state transitions are not tied to fixed temporal intervals but instead triggered when the agent moves more than a predefined spatial threshold.

In this paper, we aim to validate these intuitions by characterizing the impact of decision step frequency in continuous-time goal-conditioned reinforcement learning. We provide an empirical study of the effect of decision step configurations across three research-oriented open source environments inspired by video game mechanics, and a complex fully fledged commercial video game. We show that inappropriate frequencies induce sharp failures in goal reachability and trajectory quality, with sensitivity increasing in more complex settings. We also provide practical guidance for selecting effective frequency ranges through Pareto fronts between convergence speed and trajectory optimality. Finally, we provide theoretical insights on exponential decay of the probability of reaching the goal as the decision step frequency increases, with rates depending on the dimensionality of the physical space, as proven in Appendix A.

2 Related work

The study of decision step frequency in RL is closely related to earlier notions of action persistence (or action repetition) and frame skipping. These consist of executing the same action for $k \geq 1$ consecutive frames, which corresponds to a decision step frequency of $\frac{1}{k}$ per frame. According to Kalyanakrishnan et al. (2021), action repetition was first introduced by Buckland & Lawrence. (1994) to improve agents performance. Indeed, subsequent analyses argued that frame skipping can be beneficial in as it shortens the effective task horizon, thereby easing optimization and exploration (Andrychowicz et al., 2021). The approach became standard practice with DQN (Mnih et al., 2013), where a default skip of $k = 4$ was adopted to reduce computational cost, allowing more training episodes to be simulated within the same runtime.

There is substantial literature analyzing frame skipping, continuous-time decision frequency and action repetition across different environments and algorithms (Andrychowicz et al., 2021; Braylan et al., 2015; Kalyanakrishnan et al., 2021). Similarly to state abstraction (and state aggregation) Abel et al. (2016), action repetition leverages the observation that states occurring in close continuous-time proximity are often similar, thereby reducing the effective combinatorial complexity of the decision process. For instance, Kalyanakrishnan et al. (2021) conducted an empirical study over multiple environments, including *Acrobot* where they reported that reducing the decision step frequency from 5 Hz to 1 Hz improved performance. Authors also provide a theoretical examination of how frame skipping influences value-function learning, and evaluated frame skipping in a more complex domain that is *HFO*, where the value $k = 32$ emerged as optimal, albeit under a limited sweep of five values of k and a single random seed per setting.

Beyond fixed settings, several works attempt to learn the frame-skipping parameter jointly with the policy (Ni & Jang, 2022). FiGaR (Sharma et al., 2017) predicts the number of steps for which an action should be repeated, while TempoRL (Biedenkapp et al., 2021) conditions the repetition count on the selected action, yielding a more adaptive schedule. TACOS (Treven et al., 2024) optimizes policies that predict both control actions and their application durations to minimize costly system interactions. However, these approaches introduce additional architectural and optimization complexity and some have shown mixed performance (Kalyanakrishnan et al., 2021).

On the practical side, modern RL libraries tend to expose a fixed decision step frequency (or action-repeat/frame skipping) as a configurable setting (de Lazcano et al., 2024; Juliani et al., 2020; Malagón et al., 2025). Prior work has shown that identifying an appropriate range for this parameter can yield better-balanced agents with improved performance and stability (Khan et al., 2019; Yildiz et al., 2021; Tallec et al., 2019), and even provided theoretical insights on how fixed temporal discretization affects learning (Jia & Zhou, 2022). However, these studies focus on standard dense-rewards RL benchmarks. We focus on goal-conditioned settings, in which these issues are amplified. Indeed, due to sparse success signals, the agent receives no informative gradient signal until the goal

is first encountered. Conversely, even at high frequencies, dense-reward contexts may still provide relevant gradient feedback. We provide further justification through numerical comparisons of the impact of decision step frequency in dense versus goal-conditioned settings in Appendix B.

Despite the presence of this parameter in modern libraries, RL practitioners working with complex continuous environments often continue to select the decision step frequency manually. For example, Hildisch et al. (2025) train a real-world autonomous-driving agent with a decision step frequency of 10 Hz, but then change it to 15 Hz at inference time with minimal justification. In the video game context, Gordillo et al. (2021), who employ RL for bug discovery in large game maps, break the temporal interpretation of decision step frequency by defining a new state only when the agent is more than 5 m away from any previously visited location, which is an empirically chosen threshold. In these settings, where a discrete-action RL algorithm is deployed in a continuous-time environment, it is more appropriate to reason in terms of decision step frequency rather than frame skipping, as the agent must sense and respond to events at real-time frequencies (Srinivas et al., 2017). For these reasons, and to simplify subsequent notation, we adopt the decision step frequency formulation throughout this paper. A detailed taxonomy of these distinctions is provided by Ni & Jang (2022).

To our knowledge, no prior work characterizes how the decision step frequency affects goal reachability, trajectory optimality and convergence speed in continuous-time goal-conditioned RL.

3 Empirical Evaluation: Decision Step Frequency Effects in a Continuous-Time GCRL Navigation Task within Video Games

In this section, we conduct an empirical study evaluating the influence of the decision step frequency in four environments, three research-oriented open source environments resembling or inspired by video games and a complex, fully fledged commercial video game.

3.1 Context

We consider an RL agent evolving in an environment that is a continuous physical space Q, i.e. the video game. The agent interacts with the environment at discrete decision times separated by a fixed decision interval, allowing us to define the decision frequency ν. Between two decision times, the chosen action is kept constant and applied continuously to the system dynamics.

Therefore, for any given choice of ν, on top of the physical space, the interaction process is modeled as a partially observable Markov decision process (POMDP) defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}_\nu, R, \mathcal{O}, \gamma, H_\nu \rangle$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ the action space, $\mathcal{T}_\nu$ the transition function induced by the application of action at frequency ν, R the reward function, $\mathcal{O}$ the observation space, $\gamma \in [0, 1]$ the discount factor and H_ν the episode horizon measured in decision steps.

To isolate the impact of ν, we take the maximum number of decision steps H_ν to be linear in ν. By taking $H_\nu := C\nu$, where $C \geq 1$ is a fixed constant which depends only on the physical space Q; the total duration of the experiment (as seen from the physical space) is a constant. Indeed, it is equal to the interval duration multiplied by the number of decision steps, which is precisely equal to C.

The physical space and the state space are related through a projection map $\phi : \mathcal{S} \to Q$ extracting the task-relevant variables such that the full state $s \in \mathcal{S}$ includes all information required to ensure the Markov property of the POMDP. On the other hand, we define the reachable set as all possible locations in the continuous physical space when starting at position $x_0 \in Q$, which corresponds to the union of all locations that can be reached at each step of the any episode:

$$\mathcal{R}_Q(x_0, H_\nu) = \bigcup_{t=0}^{H_\nu} \left\{ \phi(s_t) \in Q \mid s_0 \in \phi^{-1}(x_0), s_{k+1} \in \mathcal{T}_\nu(s_k, a_k), a_k \in \mathcal{A}, k = 0, \ \ldots, \ t-1 \right\} \tag{1}$$

A visual illustration of $R_Q = R_{\mathbb{R}^2}[(0, 0), 2]$ is shown in Figure 1 to aid interpretation.

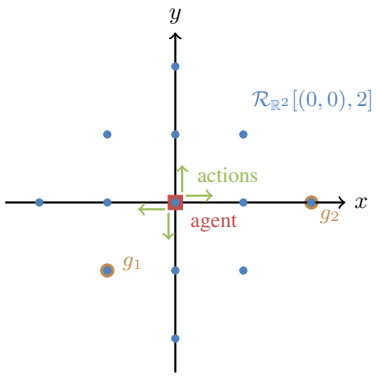

Figure 1: Example of $R_{\mathbb{R}^2}\left[(0,0),2\right]$ with $\mathcal{G} = \{g_1, g_2\}$ and axis-aligned translation actions

We consider a goal-conditioned setting in which the agent is tasked with reaching a specified goal in a goal space $\mathcal{G} \subseteq \mathcal{R}_Q(x_0, H_\nu)$, where the reward function is a sparse reward of the form:

$$R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R},\ R(s, a, s') = \omega_1 \mathbb{1}_{\phi(s') \in \mathcal{G}} - \omega_2 \tag{2}$$

where $(\omega_1, \omega_2) \in \mathbb{R}_+^2$ represent respectively the goal reward and time penalty weights.

3.2 Experimental Setting

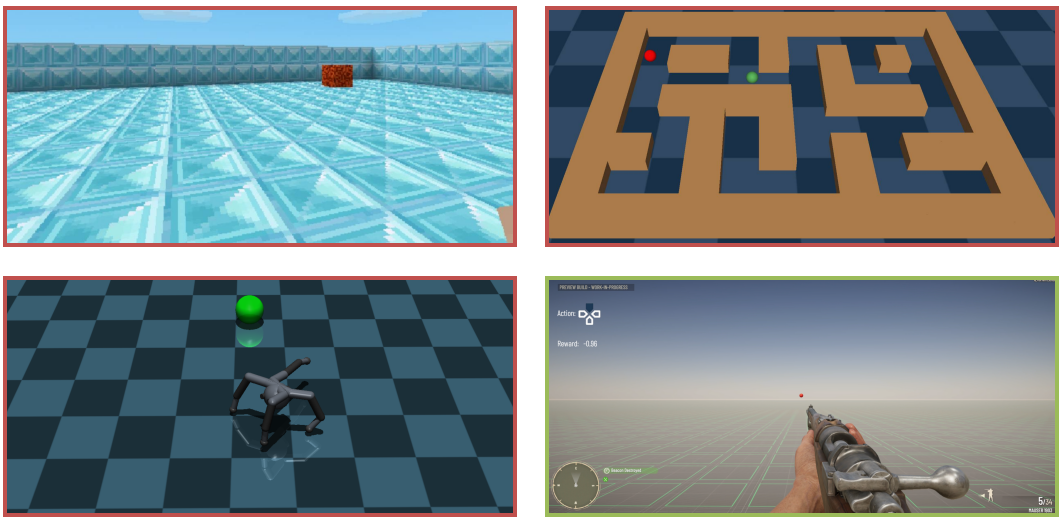

Figure 2: Four considered environments, from top left to bottom right, red borders: research-oriented environments (*Craftium:Room*, *PointMaze:Large*, and *ARC-RL:Leaper*); green border: the commercial game environment (*Gallipoli*).

Our experimental setting consists of a similar goal-reaching task across four environments shown in Figure 2, *Craftium:Room* (Malagón et al., 2025) inspired by Minecraft, *PointMaze:Large* (de Lazcano et al., 2024) which resembles Super Monkey Ball, *ARC-RL:Leaper* (Romeo & Bagdanov, 2026) inspired by Arc Raiders and finally, the commercial video game *Gallipoli*, made by Black-Mill Games. Across all environments, we maintain a controlled setup by fixing all hyperparameters and the underlying reinforcement learning architecture to isolate the impact of the auxiliary loss integration. We employ a PPO (Schulman et al., 2017) with the reward function defined in Equation 2, where $\omega_1 = 1$ and $\omega_2 = \frac{1}{C}$, ensuring that an episode in which the agent never reaches the goal yields a total return of -1. To demonstrate that our findings are algorithm-agnostic and fundamentally relate to the early stages of the training process, we provide comparable evaluations on

PointMaze:Large using Soft Actor-Critic in Appendix C. We further present these environments by increasing task difficulty:

In *Craftium:Room*, the agent operates with four discrete actions: no operation, move forward, and rotate clockwise or counterclockwise. Its objective is to learn to reach a randomly spawned red cube within an empty room using only RGB observations in $H_\nu = 2000\nu$ steps.

In *PointMaze:Large*, a green point-mass must reach a randomly located red sphere in a maze in $H_\nu = 6000\nu$ steps, using a linear horizontal force as action and its own position together with that of the goal as observation.

In the *ARC-RL:Leaper* environment, the agent controls a quadrupedal robot trying to reach a green target sphere randomly positioned 2m from the origin within of $H_\nu = 1000\nu$ steps. Observations include the robot's linear and angular velocities, height and orientation, joint positions and velocities, contact forces, and the vector to the goal.

Ultimately, we employ a complex, fully fledged commercial video game *Gallipoli*, to extend our study to a highly challenging environment. We use ML-Agents framework (Juliani et al., 2020) to integrate RL and configure the decision step parameter. In a planar environment, the agent moves at 3 m/s and has to reach a target randomly placed at 50 meters in $H_\nu = 500\nu$ steps. Observations comprises a semantic 3D map as introduced by Sestini et al. (2024) of resolution $64 \times 3 \times 64$, the vector to the goal, and the angle between the agent's facing direction and this vector. The action space includes moving forward, backward, left, or right, and rotating clockwise or counterclockwise. We provide further (non-central) details on environments and implementations in Appendix D.

To quantify the impact of the decision step frequency on learning dynamics and performance, we evaluate three complementary metrics: **Evaluation Success Rate** computed as the proportion of 100 evaluation episodes in which the agent reaches the goal; **Average Evaluation Return** over 100 evaluation episodes, capturing both goal-reaching success and trajectory efficiency; and **Normalized Convergence Step**, obtained by dividing the convergence step for each configuration by the largest convergence time within the corresponding environment. All episode-horizon values are chosen to ensure attainability under any considered decision step frequency. PPO convergence is determined by detecting performance plateaus using the difference between the average of episodic returns over 100 training episodes and the average evaluation return being lower than 5e-2. For each considered decision step frequency, 10 trainings were done using different random seeds.

3.3 Experiments and Discussion

We report the aforementioned metrics as boxplots for each environment in Figure 3, Figure 4, Figure 5 and Figure 6. We adopted a coarse-to-fine sampling of ν to ensure the results are well-represented and clearly distributed on the logarithmic scale of our figures.

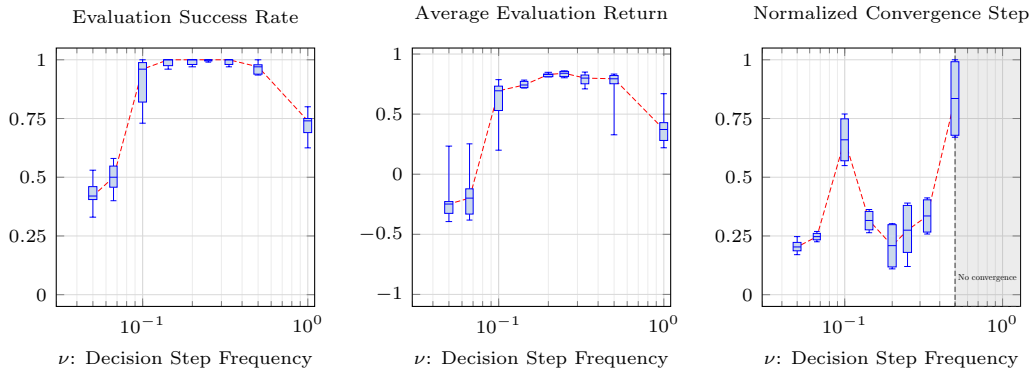

Figure 3: Empirical evaluation on the *Craftium:Room* environment

Across all environments, excessively high and excessively low decision-step frequencies lead to greater variability and a sharp decline in evaluation success rates, with this effect becoming more pronounced in more complex tasks. In the simplest setting, *Craftium:Room*, optimal success rates occurs for $\nu \in [1.4\text{e-}1, 3.3\text{e-}1]$. The more challenging maze navigation in *PointMaze:Large* exhibits a narrower effective range of $\nu \in [5\text{e-}3, 4\text{e-}2]$. Eventually, in both *ARC-RL:Leaper* and *Gallipoli*, performance steeply declines as soon as the frequency is beyond respectively 2e-2 and 4e-2.

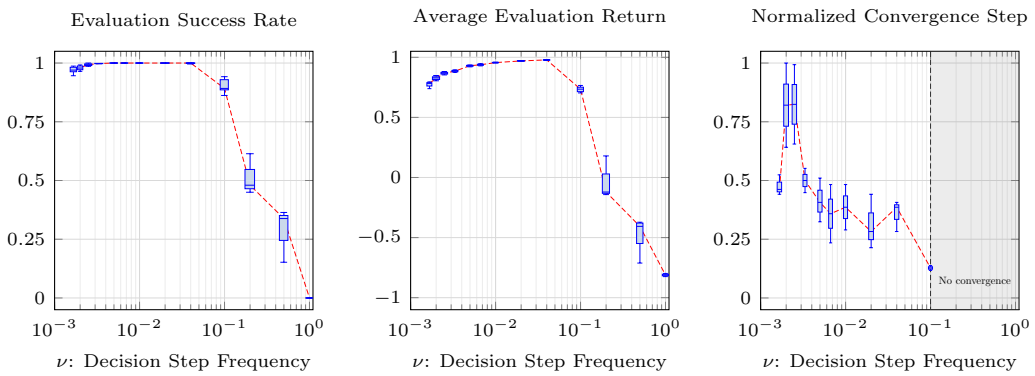

Figure 4: Empirical evaluation on the *PointMaze:Large* environment

Even in regimes where the median evaluation success rate reaches 1, inadequate decision step frequency can still produce suboptimal evaluation returns, indicating inefficient trajectories. This behavior is consistent across all four environments: in *Craftium:Room* for $\nu = 3.3\text{e-}1$ (Figure 3); in *PointMaze:Large* for $\nu \in [3.3\text{e-}3, 4\text{e-}2]$ (Figure 4); in *ARC-RL:Leaper* for $\nu \in [1.6\text{e-}3, 2\text{e-}3]$ (Figure 5); and in *Gallipoli* for $\nu \in \{1.2\text{e-}2, 6.6\text{e-}2\}$ (Figure 6).

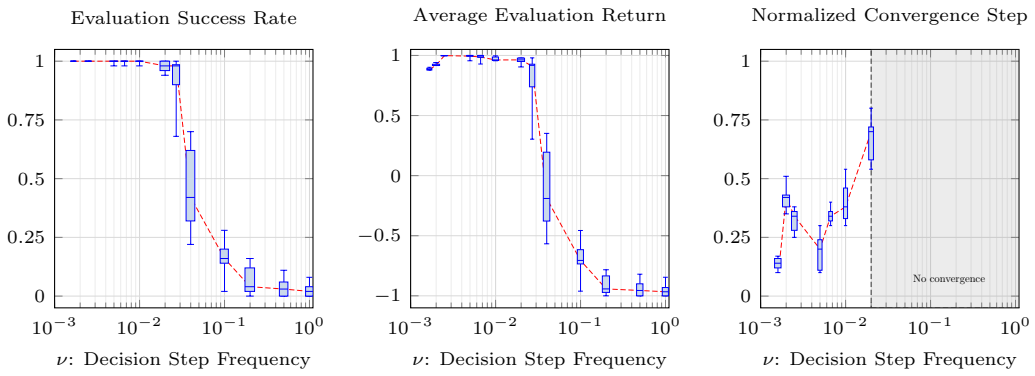

Figure 5: Empirical evaluation on the *ARC-RL:Leaper* environment

The four environments exhibit distinct patterns in how decision step frequency affects convergence speed. Because this parameter simultaneously influences convergence speed and average evaluation return, even when the median success rate (over 10 training runs) equals 1, we further examine this trade-off using Pareto fronts, shown for each experiment in Figure 7. These Pareto curves allow one to restrict the set of relevant decision step frequencies to a narrower interval, enabling practitioners to select values according to their priorities: faster convergence, higher-quality trajectories, or a balanced compromise between the two. The resulting candidate subsets are $\nu \in \{0.25, 0.33\}$ for *Craftium:Room*, $\nu \in \{0.02, 0.04, 0.1\}$ for *PointMaze:Large*, $\nu \in \{0.0016, 0.005, 0.025\}$ for *ARC-RL:Leaper*, and $\nu \in \{0.02, 0.027\}$ for *Gallipoli*.

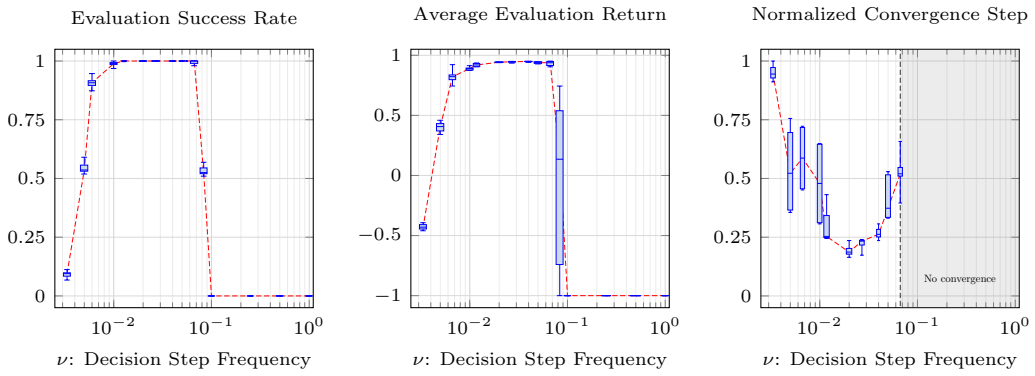

Figure 6: Empirical evaluation on the *Gallipoli* environment

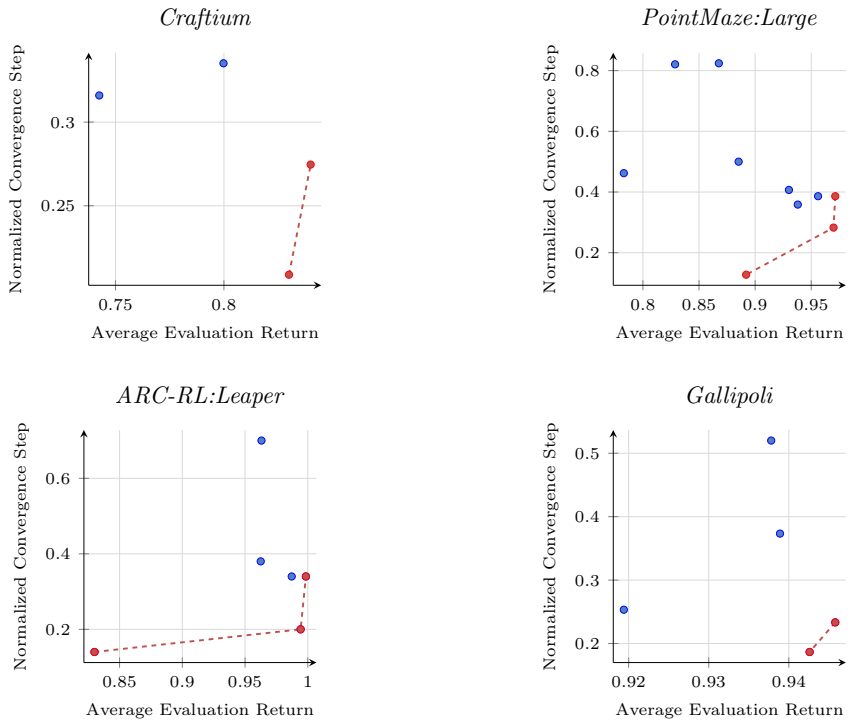

Figure 7: Pareto front (in red) of the normalize convergence step and the average evaluation return when the median of success rate over 10 training is 1, for each environment

4 Theoretical Analysis of Decision Step Frequency in Continuous-Time GCRL

Across all three investigated tasks, we observe a significant degradation in agent performance as ν increases. This decline becomes increasingly sharp as tasks complexity grows. We hypothesize that this performance collapse follows decays at least exponentially fast relative to ν. In this section, we complement our empirical evaluation with a theoretical analysis in which, under acceptable assumptions, the probability of reaching the goal decays exponentially as the decision step frequency increases, with rates depending on the dimensionality of the physical space. This theoretical bound remains directly correlated with empirical performance during the initial exploration phase, as the random walk assumption holds until the agent encounters the goal for the first time and receives an informative reward signal.

4.1 Additional Assumptions

We take a first assumption on the effect of the actions, so that the number of paths that lead the agent to the goal becomes tractable.

Assumption 1. The continuous physical space Q is in bijective correspondence with $\mathbb{R}^d$, with $d \in \mathbb{N}$. Each action induces a fixed-length translation along one of d pairwise orthogonal directions. Therefore, $\mathcal{R}_Q(x_0, H_\nu)$ is in bijective correspondence with $\{x \in \mathbb{Z}^d \mid \|x\|_1 \leq H_\nu\}$.

Assumption 2. The actions translate the agent by a distance equal to the interval duration $1/\nu$. The goal is located at $(\nu, \ldots, \nu) \in \mathcal{R}_Q(x_0, H_\nu)$, but our results extend to any position whose ℓ^1 norm is $d\nu$, where ν is a natural number.

Note that under Assumption 2., the agent needs d time units, as seen from the physical space, to reach the goal, independently of the value of ν. Indeed, the agent needs to take at least $d\nu$ decisions, with an interval duration of $1/\nu$. This can be compared to the total time spent by the agent in the physical space, which we set to C time units in subsection 3.1.

Assumption 3. Recall that C is a constant of the POMDP, that depends only on the physical space Q. Under Assumption 1., we set it as $C_d := C_1 d$.

We made this assumption to analyze the impact of ν and d. We evaluate the difficulty for the agent to reach the goal while enforcing that the ratio between the total duration of the experiment over the minimum duration to reach the goal is a constant, here equal to C_1.

4.2 Exponential Decay of Goal Reachability for High-Frequency Decision Steps

We show that, under the assumptions of subsection 4.1, the proportion of paths that reach the goal at least once decreases exponentially fast when ν increases for a fixed d. Furthermore, this proportion also decreases exponentially fast when d increases for a fixed, sufficiently large ν. We denote by $p(\nu, d)$ the proportion of paths that reach the goal at least once.

Theorem. (Proven in Appendix A)
The proportion of paths that reach the goal at least once satisfies:

$$p(\nu, d) \leq \left(2 \exp\left(\frac{-\nu}{2C_1}\right)\right)^d \tag{3}$$

Note that the proportion of paths that reach the goal is equal to the probability that a simple symmetric random walk starting at x_0 reaches the goal. In GCRL, the policy receives no informative gradient signal prior to the initial goal encounter. Thus, following random initialization, before the first success, the policy behaves essentially as a random walk. This supports the fact that the probability of reaching the goal at least once is very low when ν is high.

5 Conclusion

We formalized the decision step frequency problem in continuous-time, goal-conditioned RL and jointly characterized its impact through theory and experiments. Empirically, across three open-source tasks and a complex commercial video game, we showed that both excessively low and excessively high frequencies lead to failures in success rate and degraded trajectory quality, including in cases where median success remains 1. We also argued that goal reachability decays exponentially as the decision-step frequency increases, with rates determined by the dimensionality of the physical space. Taken together, these findings establish decision-step frequency as a critical hyperparameter for deploying reinforcement learning in production environments such as video games, with optimal performance achieved only within a narrow, task-dependent intermediate range.

A Proof of the Exponential Decay of Goal Reachability for Increasing Decision Steps

In this appendix, we provide below a proof for Equation 4.2. As explained in subsection 4.1, the reachable set of all possible positions for the RL agent in the continuous physical space R_Q is bijective with $\{x \in \mathbb{Z}^d \mid \|x\|_1 \leq H_\nu\}$, where $d \in \mathbb{N}$ is the dimension of the physical space. For the sake of simplicity, we shall assume in this section that the reachable set is $\{x \in \mathbb{Z}^d \mid \|x\|_1 \leq H_\nu\}$. Without loss of generality, we assume that x_0 is the origin.

We start by proving the following lemma, which corresponds to the **Theorem** in subsection 4.2 when taking $d = 1$. We use this result in a second step to generalize to any dimension. We use well-known results from discrete random walk theory. In particular, we rely on the fact that the proportion of paths that satisfy some property is equal to the probability that a simple symmetric random walk satisfies this property.

Lemma. *When $d = 1$, the proportion of paths of length n that reach a goal at distance t from x_0 at least once is at most $2\exp(-\frac{t^2}{2n})$. In particular, by taking $t = \nu$ and $n = C_1\nu$, we obtain: $p(\nu, 1) \leq 2\exp\left(\frac{-\nu}{2C_1}\right)$.*

Proof. We are interested in the proportion of paths that reach the goal. As $d = 1$, the agent evolves in $\mathbb{Z}$. In this proof, all considered paths have length n.

Let us denote by Π the number of paths that reach t at least once. Similarly, let us denote by Π_i the number of paths that reach t at least once and end at i, for i in $\mathbb{Z}$. We have $\Pi = \sum_{i\in\mathbb{Z}} \Pi_i$. Observe that if a path ends at $i \geq t$, then it has necessarily reached t at least once. Let us denote by Ψ_i the number of paths that end at i, without necessarily having reached t. Therefore, we obtain $\Pi = (\sum_{i<t} \Pi_i) + (\sum_{i\geq t} \Psi_i)$.

Using the reflection principle, we show in this paragraph that the set of paths that reach t at least once and end at $i < t$, and the set of paths that end at $2t - i$ are in bijective correspondence, thus implying $\Pi_i = \Psi_{2t-i}$ for $i < t$. Let us now consider a path that reach t at least once, and ends at $i < t$. Let us consider the first time when the path reaches t, and let us reflect all of the remaining steps. The original path went from t to $i < t$, thus the reflected path goes from t to $t + (t - i) = 2t - i$. As $t < i$, we have $2t - i > t$, which implies that the reflected path reaches t at least once. Observe that this reflection maps two different paths to two different reflected paths, and vice versa. Therefore, it is a bijection.

We have $\Pi = (\sum_{i<t} \Pi_i) + (\sum_{i\geq t} \Psi_i) = (\sum_{i<t} \Psi_{2t-i}) + (\sum_{i\geq t} \Psi_i) = (\sum_{j\geq t+1} \Psi_j) + (\sum_{i\geq t} \Psi_i)$ by setting $j = 2t - i$. We infer $\Pi \leq 2\sum_{i\geq t} \Psi_i$.

To obtain an upper bound on Ψ_i, we use a one-dimensional simple symmetric random walk. For $1 \leq k \leq n$, let X_k be a uniform random variable with range equal to $\{-1, 1\}$. We take the X_k to be independent, and $S_n := \sum_{1\leq k\leq n} X_k$ is a simple random walk on $\mathbb{Z}$. Observe that the proportion of paths that end at i is equal to the probability that S_n is equal to i. As there are n steps, there exist 2^n paths. Thus, the proportion of paths that reach t at least once is equal to $\Pi/2^n \leq 2^{1-n} \sum_{i\geq t} \Psi_i = 2\Pr(S_n \geq t)$.

We use Hoeffding's inequality to obtain an upper bound of this probability (Hoeffding, 1963). It states $\Pr(S_n - \mathbb{E}(S_n) \geq t) \leq \exp\left(-\frac{2t^2}{\sum_{1\leq k\leq n}(b_k - a_k)^2}\right)$, where $a_k \leq X_k \leq b_k$ almost surely. Here, we have $a_k = -1$, $b_k = 1$ and $\mathbb{E}(X_k) = 0$, for all $1 \leq k \leq n$. By replacing the variables in the formula, we obtain $\Pr(S_n \geq t) \leq \exp(-\frac{2t^2}{\sum_{1\leq k\leq n} 2^2}) = \exp(-\frac{t^2}{2n})$. □

Proof of the theorem in subsection 4.2. We use again the fact that the proportion of paths that reach the goal is equal to the probability that a simple symmetric simple random walk reach the goal. After $C_d\nu$ steps, let us denote by M_i the number of steps made by the random walk in the i-th coordinate. The M_i are not independent, since their sum is equal to $C_d\nu$. As the random walk is symmetric,

we have that $(M_1, \ldots, M_d)$ follows a multinomial distribution $\mathcal{M}_d(C_d\nu;\ 1/d, \ldots, 1/d)$. We denote by $S^{(i)}$ the one-dimensional random walk obtained by only considering the steps made along the i-th coordinate.

Let us denote by $P_{m,\nu}$ the probability that a one-dimensional simple symmetric random walk reaches ν at least once within m steps. Let E denote the event that our d-dimensional walk reaches the goal. Recall that the position of the goal is $(\nu_1, \ldots, \nu_d)$, such that $\sum_{i=1}^{d} |\nu_i| = d\nu$. Without loss of generality, we assume the ν_i to be non-negative.

By the law of total probability, we have $\Pr(E) = \sum_{m_1+\cdots+m_d=C_d\nu} \Pr(M_1 = m_1, \ldots, M_d = m_d) \Pr(E|M_1 = m_1, \ldots, M_d = m_d)$. Given the (M_i) values, the $(S^{(i)})$ are pairwise independent, since their only link is that they must sum up to $C_d\nu$. Note that if the agent reaches the goal at time τ, then each $S^{(i)}$ reaches ν_i at time τ. To get an upper bound on $\Pr(E)$, we use the fact that the probability that each $S^{(i)}$ reaches ν_i at time τ is less than the probability that each $S^{(i)}$ must have reached ν_i at some time $\tau_i \leq C\nu$. Thus we have $\Pr(E) \leq \sum_{m_1+\cdots+m_d=C_d\nu}(\Pr(M_1 = m_1, \ldots, M_d = m_d) \prod_{i=1}^{d} P_{m_i,\nu_i})$.

By the **Lemma** written above, we have $P_{m_i,\nu_i} \leq 2\exp(-\frac{\nu_i^2}{2m_i})$, and thus $\prod_{i=1}^{d} P_{m_i,\nu_i} \leq 2^d \exp(-\frac{1}{2}\sum_{i=1}^{d} \frac{\nu_i^2}{m_i})$. To find an upper bound, we need a lower bound on $\sum_{i=1}^{d} \frac{\nu_i^2}{m_i}$. By Sedrakyan's inequality (Sedrakyan, 1997), we have $\sum_{i=1}^{d} \frac{\nu_i^2}{m_i} \geq \frac{(\sum_{i=1}^{d} \nu_i)^2}{\sum_{i=1}^{d} m_i} = \frac{d^2\nu^2}{C_d\nu} = \frac{d\nu}{C_1}$.

We obtain $\prod_{i=1}^{d} P_{m_i,\nu_i} \leq 2^d \exp(-\frac{d\nu}{2C_1}) = (2\exp(-\frac{\nu}{2C_1}))^d$. We can conclude with $\Pr(E) \leq (2\exp(-\frac{\nu}{2C_1}))^d \sum_{m_1+\cdots+m_d=C\nu} \Pr(M_1 = m_1, \ldots, M_d = m_d) = (2\exp(-\frac{\nu}{2C_1}))^d$. □

B Empirical evaluation in a dense-reward setting

In this appendix, we further justify the relevance of evaluating goal-conditioned setting particularly in contrast to dense-reward by comparing performance from our primary study in *PointMaze:Large* (this environment was selected for this evaluation due to its fast simulation) against that of a variant employing a standard dense-reward signal based on normalized distance (Ahlberg et al., 2025).

Early insights in Figure 8 show that the decision step frequency impact over evaluation success rate is mitigated by the presence of a reward signal at every decision interval. Notably, at $\nu = 1$, the dense-reward configuration achieves a median success rate of 26%, representing an improvement over the total failure observed in the sparse-reward setting where the agent receives no informative gradient signal until the goal is first successfully reached. Across all ν values tested, results in the dense context consistently match or exceed their sparse counterparts, even at lower frequencies where the dense reward allows to slightly mitigate the decline in success rates.

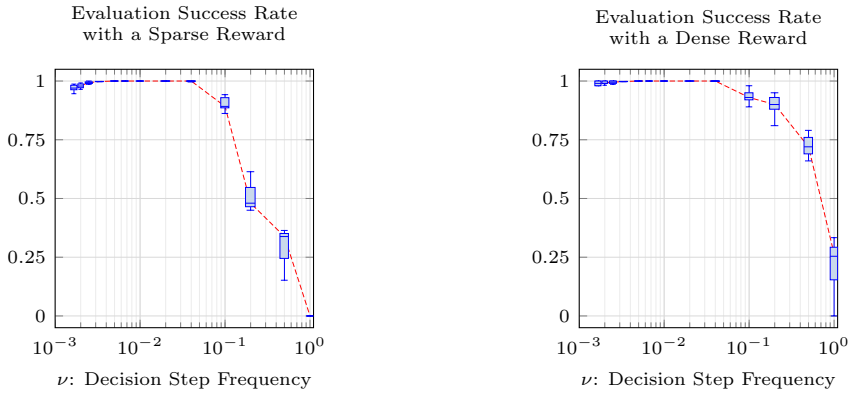

Figure 8: Empirical evaluation on the *PointMaze:Large* environment in dense *vs* GCRL settings

C Empirical evaluation using SAC

In this work, we restrict our analysis to PPO agents in order to isolate the impact of the decision step frequency. Our work focuses on showing that inappropriate selection of this parameter can constitute a bottleneck that prevents efficient learning, as explained in section 4. Nevertheless, to mitigate the algorithm-specific nature of this analysis, we reproduce our experiment using Soft Actor-Critic (SAC) on the *PointMaze:Large* environment, again selected for its fast simulation speed.

For these experiments, we employ the default SAC implementation from stable-baselines3 (Raffin et al., 2021), using the hyperparameters reported in Table 1. We compare evaluation return rate using SAC and PPO in Figure 9.

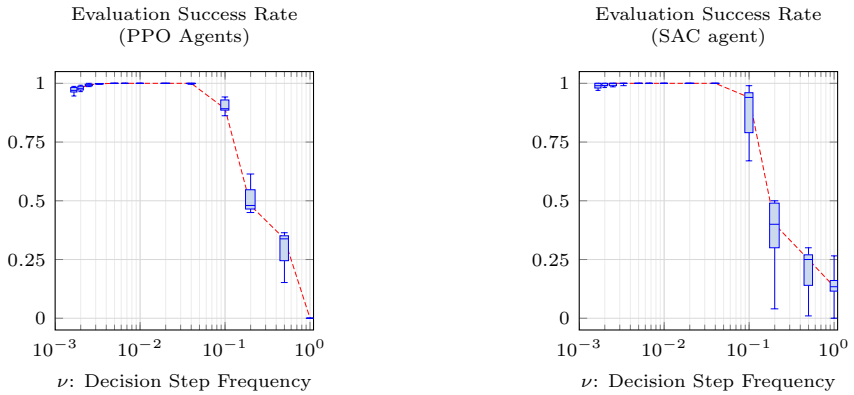

Figure 9: Empirical evaluation on the *PointMaze:Large* environment using a SAC *vs* PPO agent

While SAC achieves a higher median success rate of 0.2 at $\nu = 1$, the high variance is most likely validating the impact of the off-policy replay buffer. Given that the probability of encountering a sparse goal decays exponentially at high frequencies (section 4), such successful trajectories are exceptionally rare. Whereas on-policy gradient updates in PPO suffer from sparse signals, SAC captures and isolates these rare events within its buffer, allowing the agent to continuously sample and exploit them. At low frequencies, we observe a similar but mitigated reduction in performance.

Table 1: Parameters of the SAC Empirical Evaluation on *PointMaze:Large*

PARAMETER	**VALUE**
Batch, Replay Buffer size	256, 1e6
Learning rate	3e-4
Discount factor	0.99
Target smoothing coefficient	5e-3
Horizon H_ν	500ν steps
Gradient step	1
Timestep starting training	100 steps
Total timesteps for training	3e5 steps
Target update interval	1
Tested ν values	{5e-3, 6.6e-3, 1e-1, 1e-1, 1.4e-1, 2e-1, 2.5e-1, 3.3e-1, 5e-1, 1}

D Environments and implementations details

In this appendix, we first provide additional details on PPO implementations, environments and training parameters. All experiments were performed on an AMD Ryzen Threadripper PRO 7965WX 24-Core CPU and a NVIDIA RTX 6000 Ada Generation GPU.

D.1 *PointMaze:Large* experiment

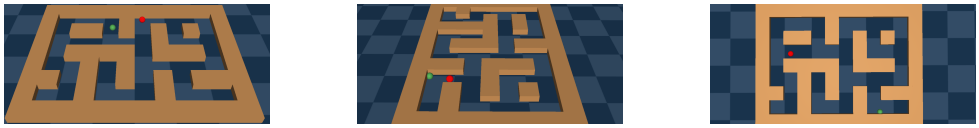

Figure 10: Additional *PointMaze:Large* environment screenshots

We choose to employ a PPO agent using stable-baselines3 (Raffin et al., 2021) with the default `MultiInputPolicy`, since this environment returns a dictionary as the observation space. This architecture flattens the observation data before using MLP extractors and finally separate actor and value heads for policy and state-value predictions, as shown in Figure 11.

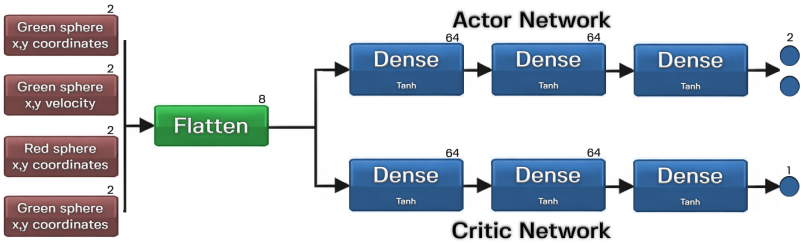

Figure 11: Diagram of the MultiInputPolicy used in the *PointMaze:Large* environment

We also modify the reward signal, which normally assigns 0 at each step and 1 upon reaching the goal, to the reward specified in subsection 3.2, thereby introducing a time penalty. Training is performed using backpropagation with the Adam optimizer. We provide additional hyperparameters in Table 2. Most values have been kept as default to focus on the decision step frequency analysis.

Table 2: Parameters of the Empirical Evaluation on *PointMaze:Large*

PARAMETER	VALUE
Batch, Buffer size	128, 2048
Learning rate	3e-4
Entropy, Critic loss coefficient	1e-2, 0.5
Epsilon clip	0.2
GAE Lambda	0.95
Discount factor	0.99
Gradient steps	3
Horizon H_ν	6000ν steps
Total timesteps for training	3e5 steps
Tested ν values	{1.6e-3, 2e-3, 2.5e-3, 3.3e-3, 5e-3, 1e-2, 2e-2, 4e-2, 1e-1, 2e-1, 5e-1, 1}

D.2 *Craftium:Room* experiment

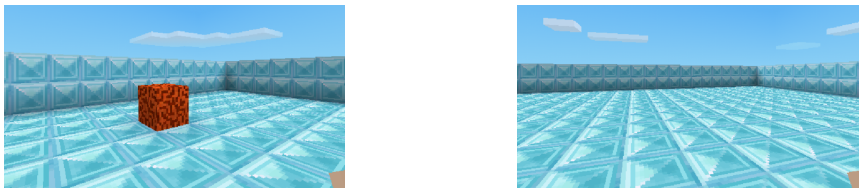

Figure 12: Additional *Craftium:Room* environment screenshots

Similarly, we employ a PPO agent using stable-baselines3 (Raffin et al., 2021) with the default `CnnPolicy`, since the observation space consists of images and we want to preserve the spatial structure of the input. This architecture consists of a Convolutional Neural Network feature extractor followed by an fully connected layer as backbone, before separate actor and value heads for policy and state-value predictions as shown in Figure 13.

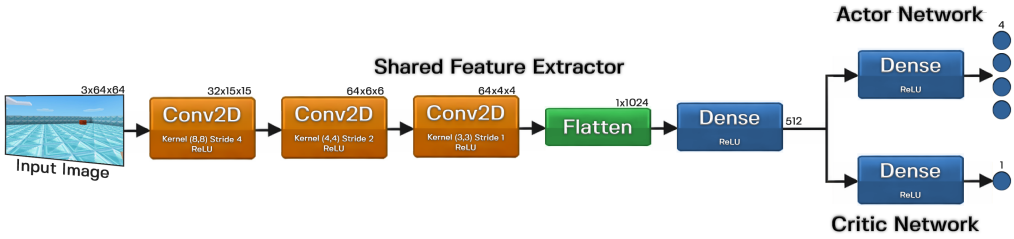

Figure 13: Diagram of the CnnPolicy used in the *Craftium:Room* environment

We modify the reward signal, which normally assigns -1 at each step, to the reward specified in subsection 3.2, thereby introducing both a time penalty and reward upon reaching the goal.

Default observation is used which corresponds to a RGB capture of the environment of resolution 64×64 pixels. We also employ Adam as optimizer and rely on the set of hyperparameters close to the previous environment except for the horizon and the epochs number, as described in Table 3. Just like the observation, and like the previous experiment, most values have been kept as default to focus on the decision step frequency analysis.

Table 3: Parameters of the Empirical Evaluation on *PointMaze:Large*

PARAMETER	**VALUE**
Batch, Buffer size	$128, 2048$
Learning rate	3e-4
Entropy, Critic loss coefficient	1e-2, 0.5
Epsilon clip	0.2
GAE Lambda	0.95
Discount factor	0.99
Gradient steps	10
Horizon H_ν	500ν steps
Total timesteps for training	3e5 steps
Tested ν values	{5e-3, 6.6e-3, 1e-1, 1e-1, 1.4e-1, 2e-1, 2.5e-1, 3.3e-1, 5e-1, 1}

D.3 *Arc-RL:Leaper* experiment

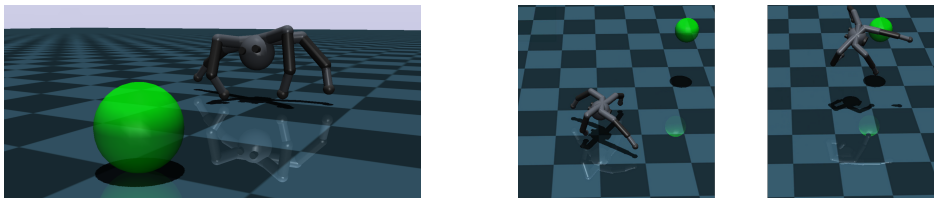

Figure 14: Additional *Arc-RL:Leaper* environment screenshots

We modified this environment in several key aspects. First, we replaced the default SAC implementation with PPO to match other experiments. Second, we drastically modified the reward signal: while the original environment employs complex reward shaping to encourage aesthetically pleasing forward movement resembling the game ARC Raiders, we utilized the sparse reward detailed in subsection 3.2. By focusing solely on goal reaching and a step-wise time penalty, the agent optimizes for task efficiency rather than movement aesthetics, as shown in Figure 14.

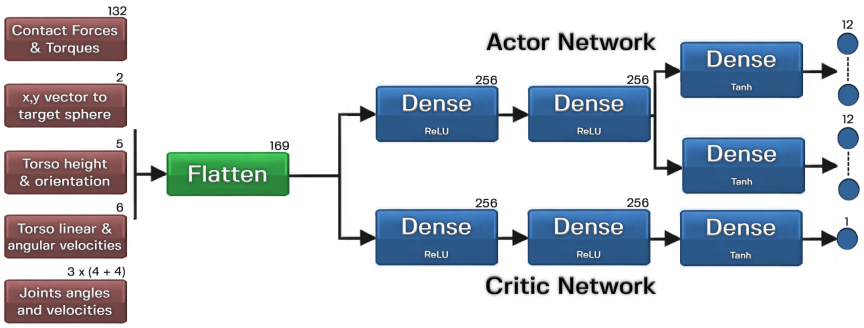

Figure 15: Diagram of the architecture used in the *Arc-RL:Leaper* environment

Maintaining the default configurations (hyperparameters specified in Table 4, the policy utilizes a dense architecture as illustrated in Figure 15. We refined the observation space by omitting the torso's planar coordinates and the phase used for trot. Consequently, the input features are restricted to 132 contact force vectors, 12 joints positions and velocities (3 per legs), the torso height and velocity; and the vector to the goal.

Table 4: Parameters of the Empirical Evaluation on *Arc-RL:Leaper*

PARAMETER	VALUE
Batch, Buffer size	$256, 2048$
Learning rate	3e-4
Entropy, Critic loss coefficient	1e-2, 0.5
Epsilon clip	0.2
GAE Lambda	0.95
Discount factor	0.99
Gradient steps	1
Horizon H_ν	1000ν steps
Total timesteps for training	1e6 steps
Tested ν values	$\{$1.7e-3, 2e-3, 2.5e-3, 5e-3, 6.7e-3, 1e-2, 2.7e-2, 2e-2, 4e-2, 1e-1, 2e-1, $1\}$

D.4 *Gallipoli* environment

Figure 16: Additional *Gallipoli* environment screenshots

The agent used in the *Gallipoli* environment is trained with PPO (Schulman et al., 2017) using ML-Agents (Juliani et al., 2020). The PPO actor network outputs two linear heads corresponding to the agent's two discrete action branches, as shown in Figure 17. The first head is a categorical distribution over movement actions and the second head is a categorical distribution over rotational actions. These two information are simultaneously transmitted and applied in *Gallipoli*, allowing for a behavior close to a real player.

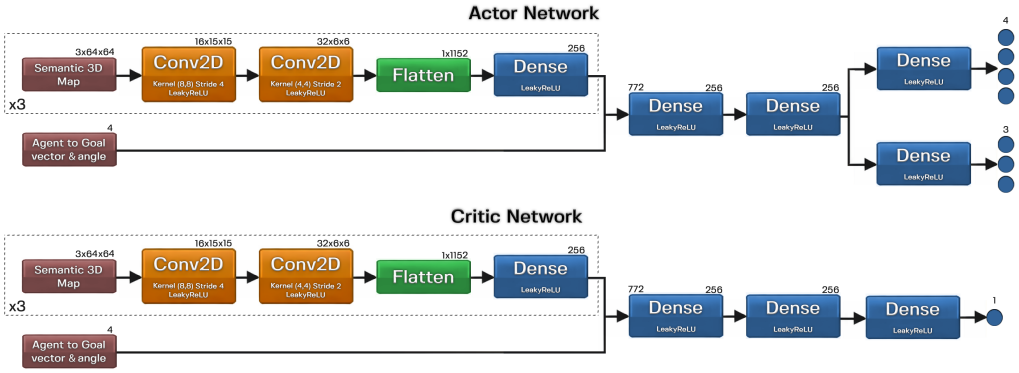

Figure 17: Diagram of the actor and critic architectures used in the *Gallipoli* environment

Training is performed using backpropagation with the Adam optimizer. The 3D semantic map uses voxels of size $0.1\text{m} \times 0.5\text{m} \times 0.1\text{m}$, chosen to match the spatial scale of the game and to ensure that the discretization captures navigation-relevant structure in the scene. Eventually, we provide additional hyperparameters in Table 5. These were chosen as usual values for RL training.

Table 5: Parameters of the Empirical Evaluation on *Gallipoli*

PARAMETER	**VALUE**
Batch, Buffer size	128, 2048
Learning rate	3e-4
Entropy, Critic loss coefficient	1e-2, 0.5
Epsilon clip	0.2
GAE Lambda	0.95
Discount factor	0.99
Gradient steps	3
Horizon H_ν	500ν steps
3D semantic map resolution	$6.4\text{m} \times 1.5\text{m} \times 6.4\text{m}$
Total timesteps for training	3e5 steps
Tested ν values	{3.3e-3, 5e-3, 6.6e-3, 1e-2, 1.2e-2, 2e-2, 2.7e-2, 4e-2, 5e-2, 5.5e-2, 6e-2, 8e-2, 1e-1, 2.5e-1, 5e-1, 1}

References

David Abel, D. Ellis Hershkowitz, and Michael L. Littman. Near optimal behavior via approximate state abstraction. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 2915–2923. JMLR.org, 2016.

William Ahlberg, Alessandro Sestini, Konrad Tollmar, Linus Gisslén, and Iolanda Leite. Exploring model-based shielding for non-player character behaviour. In *Reinforcement Learning and Video Games Workshop @ RLC 2025*, 2025. URL https://openreview.net/forum?id=en6i5l6ObB.

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Leonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters for on-policy deep actor-critic methods? a large-scale study. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=nIAxjsniDzg.

André Biedenkapp, Raghu Rajan, Frank Hutter, and Marius Lindauer. Temporl: Learning when to act. *CoRR*, abs/2106.05262, 2021. URL https://arxiv.org/abs/2106.05262.

Alexander Braylan, Mark Hollenbeck, Elliot Meyerson, and Risto Miikkulainen. Frame skip is a powerful parameter for learning to play atari. In *AAAI Workshop: Learning for General Competency in Video Games*, 2015. URL https://api.semanticscholar.org/CorpusID:194604.

Kenneth M. Buckland and Peter D. Lawrence. Transition point dynamic programming. In *Advances in Neural Information Processing Systems*, volume 6, pp. 639–646, 1994.

Aurélien Chambon and Nicolas Grelier. Comparing high entropy reinforcement learning to navmesh for automated collision bug testing. In *Reinforcement Learning and Video Games Workshop @ RLC 2025*, 2025. URL https://openreview.net/forum?id=PIWiufVV5a.

Rodrigo de Lazcano, Kallinteris Andreas, Jun Jet Tai, Seungjae Ryan Lee, and Jordan Terry. Gymnasium robotics, 2024. URL http://github.com/Farama-Foundation/Gymnasium-Robotics.

Camilo Gordillo, Joakim Bergdahl, Konrad Tollmar, and Linus Gisslén. Improving playtesting coverage via curiosity driven reinforcement learning agents. In *2021 IEEE Conference on Games (CoG)*, pp. 1–8. IEEE Press, 2021. DOI: 10.1109/CoG52621.2021.9619048. URL https://doi.org/10.1109/CoG52621.2021.9619048.

Benedict Hildisch, Edoardo Ghignone, Nicolas Baumann, Cheng Hu, Andrea Carron, and Michele Magno. Drive fast, learn faster: On-board RL for high performance autonomous racing. *Reinforcement Learning Journal*, 6:829–861, 2025.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

Yanwei Jia and Xun Yu Zhou. Policy gradient and actor-critic learning in continuous time and space: Theory and algorithms. *Journal of Machine Learning Research*, 23(275):1–50, 2022. URL http://jmlr.org/papers/v23/21-1387.html.

Arthur Juliani, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*, 2020. URL https://arxiv.org/pdf/1809.02627.pdf.

Shivaram Kalyanakrishnan, Siddharth Aravindan, Vishwajeet Bagdawat, Varun Bhatt, Harshith Goka, Archit Gupta, Kalpesh Krishna, and Vihari Piratla. An analysis of frame-skipping in reinforcement learning. *ArXiv*, abs/2102.03718, 2021. URL https://api.semanticscholar.org/CorpusID:231847020.

Adil Khan, Jiang Feng, Shaohui Liu, and Muhammad Zubair Asghar. Optimal skipping rates: Training agents with fine-grained control using deep reinforcement learning. *Journal of Robotics*, 2019(1):2970408, 2019. DOI: https://doi.org/10.1155/2019/2970408. URL https://onlinelibrary.wiley.com/doi/abs/10.1155/2019/2970408.

Mikel Malagón, Josu Ceberio, and Jose A. Lozano. Craftium: Bridging flexibility and efficiency for rich 3d single- and multi-agent environments. In *Reinforcement Learning and Video Games Workshop @ RLC 2025*, 2025. URL https://openreview.net/forum?id=o7bhrTklwQ.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL http://arxiv.org/abs/1312.5602.

Tianwei Ni and Eric Jang. Continuous control on time. In *ICLR 2022 Workshop on Generalizable Policy Learning in Physical World*, 2022. URL https://openreview.net/forum?id=BtbG3NT4y-c.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

Daniele Reda, Tianxin Tao, and Michiel van de Panne. Learning to locomote: Understanding how environment design matters for deep reinforcement learning. In *Proceedings of the 13th ACM SIGGRAPH Conference on Motion, Interaction and Games*, MIG '20, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450381710. DOI: 10.1145/3424636.3426907. URL https://doi.org/10.1145/3424636.3426907.

Carlo Romeo and Andrew D. Bagdanov. Arc-rl: A reinforcement learning playground inspired by arc raiders, 2026. URL https://arxiv.org/abs/2605.19503.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Nairi Sedrakyan. About the applications of one useful inequality. *Kvant Journal*, 97(2):42–44, 1997.

Alessandro Sestini, Linus Gisslén, Joakim Bergdahl, Konrad Tollmar, and Andrew David Bagdanov. Automated gameplay testing and validation with curiosity-conditioned proximal trajectories. *IEEE Transactions on Games*, 16(1):113–126, 2024. DOI: 10.1109/TG.2022.3226910.

Sahil Sharma, Aravind S. Lakshminarayanan, and Balaraman Ravindran. Learning to repeat: Fine grained action repetition for deep reinforcement learning. In *ICLR (Poster)*, 2017. URL https://openreview.net/forum?id=B1GOWV5eg.

A. Srinivas, Sahil Sharma, and Balaraman Ravindran. Dynamic action repetition for deep reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2017. URL https://api.semanticscholar.org/CorpusID:4753642.

Corentin Tallec, Léonard Blier, and Yann Ollivier. Making deep q-learning methods robust to time discretization. In *International Conference on Machine Learning*, 2019. URL https://api.semanticscholar.org/CorpusID:59316710.

Lenart Treven, Bhavya Sukhija, Yarden As, Florian Dörfler, and Andreas Krause. When to sense and control? a time-adaptive approach for continuous-time rl. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

Cagatay Yildiz, Markus Heinonen, and Harri Lähdesmäki. Continuous-time model-based reinforcement learning. In *International Conference on Machine Learning*, pp. 12009–12018. PMLR, 2021.