The Dark Side of Rich Rewards: Understanding and Mitigating Noise in VLM Rewards

Sukai Huang[1], **Shu-Wei Liu**[2], **Nir Lipovetzky**[1] **and Trevor Cohn**[1*]

[1]School of Computing and Information Systems, The University of Melbourne, Australia
[2]Max Planck Institute for the Physics of Complex Systems, Germany

Abstract

While Vision-Language Models (VLMs) are increasingly used to generate reward signals for training embodied agents to follow instructions, our research reveals that agents guided by VLM rewards often underperform compared to those employing only intrinsic (exploration-driven) rewards, contradicting expectations set by recent work. We hypothesize that false positive rewards – instances where unintended trajectories are incorrectly rewarded – are more detrimental than false negatives. We confirmed this hypothesis, revealing that the widely used cosine similarity metric is prone to false positive estimates. To address this, we introduce BIMI (**Bi**nary **M**utual **I**nformation), a novel reward function designed to mitigate noise. BIMI significantly enhances learning efficiency across diverse and challenging embodied navigation environments. Our findings offer a nuanced insight of how different types of reward noise impact agent learning and highlight the importance of addressing multimodal reward signal noise when training embodied agents.

1 Introduction

Natural language instructions are increasingly recognized as a valuable source of reward signals for guiding embodied agents to learn complex tasks. In particular, a growing trend in embodied agent learning involves using vision-language models (VLMs) for reward modeling. This approach measures the semantic similarity – often quantified by cosine similarity – between the embedding representations of an agent's behaviors (i.e., past trajectories) and the provided instructions, all within the same embedding space (Kaplan et al., 2017; Goyal, Niekum, and Mooney, 2019, 2020; Du et al., 2023; Rocamonde et al., 2024; Wang et al., 2024).

However, we observed that embodied RL agents trained with VLM rewards, while effective in simplified settings, often struggled with tasks involving complex dynamics and longer action horizons[1]. This is evident in several recent works – for instance, Goyal, Niekum, and Mooney (2019) reported the effective use of VLM rewards in Montezuma's Revenge, a notoriously challenging Atari game. However, we observed that this success was confined to individual subtasks and the agent struggled when attempting to scale up to the full game. Similarly, Du et al. (2023) demonstrated

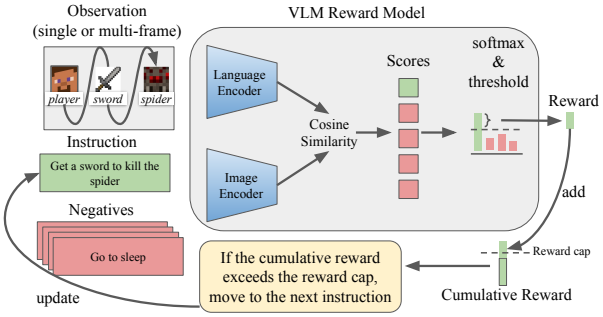


Figure 1: Illustration of embodied RL agents using VLM reward model

impressive performance of VLM rewards in guiding agents within a 2D survival game. However, their study was conducted in a modified environment with a reduced observation and action space using internal game state information and manually defined macro actions (see Appendix B). Consequently, when we tested their methods in the original, unmodified environment, we found that their agent's performance did not exceed that of agents using only *intrinsic* (exploration-driven) rewards.

The consistent underperformance of VLM rewards, **particularly their unexpected failure to outperform intrinsic rewards**, raised concerns about their reliability. This discrepancy, where VLM rewards underperformed contrary to their perceived potential, prompted us to investigate the underlying causes of this performance gap.

Our findings indicate that **noisy reward estimates** in VLMs are a key factor contributing to poor learning efficacy. Specifically, our analysis centered around two classes of noise: *false positives*, which involve rewarding unintended trajectories and can mislead the agent into reinforcing suboptimal behaviors; and *false negatives*, which occur when correct trajectories are not rewarded, thus providing little feedback and reintroduce the reward sparsity issue. We posit that false positive rewards are not only more prevalent but potentially more detrimental to the learning process than false negatives, a hypothesis supported by our empirical and theoretical findings. Among various sources of reward noise, our study particularly investigates the **approximation er-**

[*]Now at Google DeepMind
[1]Code available at https://shorturl.at/3uVGP

rors within the commonly used cosine similarity metric. We examine how these errors generate false positive rewards, which in turn hinder learning.

To this end, we propose a novel reward function, BIMI (**Bi**nary **M**utual **I**nformation Reward). It uses binary signals to directly reduce the occurrence of false positives and incorporates a mutual information term to prevent overfitting to noisy signal sources. Our experiments demonstrate that BIMI significantly improves the learning efficacy of instruction-following agents trained with VLM rewards across various complex, *long-horizon* tasks in embodied environments. We validate the effectiveness of BIMI in both *Markovian* and *non-Markovian* reward settings, showcasing its robustness and generalizability across different types of reward models.

2 Related Work

We provide a brief overview of the related work for this study. For detailed ones, please refer to Appendix B.

VLMs as Reward Models. Despite the rising use of VLM-based reward models in embodied RL (Kaplan et al., 2017; Goyal, Niekum, and Mooney, 2019, 2020; Du et al., 2023; Rocamonde et al., 2024; Wang et al., 2024, 2019; Shridhar et al., 2022; Mahmoudieh, Pathak, and Darrell, 2022), many works focus on proof-of-concept results in simplified environments, sidestepping the challenges posed by noisy reward signals. Critical issues such as reward noise and its effect on policy convergence remain largely understudied. In particular, some approaches sidestep noise by leveraging internal state information or predefined action macros (Du et al., 2023; Wang et al., 2023). However, these methods struggle to scale to real-world applications due to their reliance on environment-specific priors, which lack generalizability in dynamic, unstructured settings.

Pessimistic Value Estimation. The RL community has demonstrated that pessimism under uncertainty improves robustness in settings like offline RL optimization (Uehara and Sun, 2021; Bai et al., 2024). These methods conservatively estimate state values to avoid overoptimism in poorly explored or uncertain regions. However, such techniques are rarely applied to settings with learned VLM-based rewards, **where noise arises not from environmental uncertainty but from misaligned or overconfident reward signals**. Moreover, these methods remain largely alien to recent studies of VLM reward + RL embodied agent research. Our work bridges this gap by applying pessimism to mitigate learned reward noise, addressing a critical yet overlooked challenge in this emerging field.

Optimistic Exploration in Online RL. Pessimistic estimates in RL are mostly an offline phenomenon; online RL instead typically embraces optimism under uncertainty to drive efficient exploration. Classic model-based schemes such as RMax initialize all unknown state-action pairs with the maximum possible reward, thereby provably encouraging the agent to visit under-explored regions of the MDP (Brafman and Tennenholtz, 2002). In the model-free regime, intrinsic reward methods measure the surprise of an agent's actions, such as the prediction error of a forward model (Burda et al., 2018; Wan et al., 2023), or the novelty of state visits (Zhang et al., 2021). Our work investigates the integration of pessimistic, VLM-based reward signals with optimistic intrinsic rewards in an online embodied environment.

Reward Signal from Human Preference. Recent work on RL from Human Feedback (RLHF) (Ouyang et al., 2022) also leverage expert preference as a reward signal. However, our work differs in key aspects. Unlike RLHF's focus on textual outputs, our approach involves evaluating cross-modal similarities between visual and textual data in environments requiring long-horizon decision-making and frequent embodied interactions, a domain not typically covered by RLHF.

Mitigating Misspecified Rewards. Prior works proposed mitigating false positive rewards by training a parallel exploration policy to escape local optima caused by misspecified rewards (Ghosal et al., 2022; Fu et al., 2024). In contrast, we propose a novel reward function that directly penalizes likely false positive reward signals during training. We further show that our method complements exploration-based methods and achieves superior performance when combined.

3 Formal Problem Statement

We frame our task as an MDP defined by a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, s_0, r^e, \gamma \rangle$, where $\mathcal{S}$ represents a set of states $s \in \mathcal{S}$, $\mathcal{A}$ represents a set of actions $a \in \mathcal{A}$, and $\mathcal{P}(s'|s,a)$ describes the dynamics of the environment. $s_0 \in \mathcal{S}$ is the initial state and $\gamma \in (0,1)$ is the reward discount factor. $r^e(s,a)$ is the environmental reward function. An agent's trajectory is a sequence of states and actions $\tau_t = \langle s_0, a_0, \ldots, s_t \rangle$.

In this work, we focus on a sparse reward setting, where the agent receives a $+1$ reward only when reaching goal states $S_G \subset \mathcal{S}$, and 0 otherwise, with $|S_G| \ll |\mathcal{S}|$. This sparse reward setting motivates the use of expert instructions and VLMs to provide auxiliary reward signals for more effective RL. Here, it's crucial to note that by VLM, we specifically mean encoder-based VLMs, as opposed to generative VLMs (see Appendix M). For conciseness, we will simply use 'VLM' to refer to encoder-based VLMs.

Specifically, we have a walkthrough L that breaks down a complex task into n expert-defined sub-tasks, each represented by a natural language instruction that is not necessarily atomic and can encompass multiple finer sub-goals ($L = \{l_1, l_2, \ldots, l_n\}$). By following these sequential instructions, the agent can navigate from the initial state towards the goal states. A dedicated *non-Markovian* VLM-based reward model[2] $r^v(\tau_t, l_{m(t)})$ is used to assess how well the agent's trajectory at current time t fulfills an instruction sentence $l_{m(t)}$. For the detailed mechanism and the pseudo-code of the existing VLM reward + RL algorithm, please refer to Appendix C. Note that our proposed BIMI reward function uses a different mechanism to determine if instruction $l_{m(t)}$ is completed at time t (explained shortly in Section 7). The

[2]The use of *non-Markovian* reward functions in MDP has been well-established, particularly through the work on reward machines (Icarte et al., 2018; Corazza, Gavran, and Neider, 2022). For a complete evaluation, we also tested *Markovian* version of VLM reward function (Pixl2R) in our experiments.

VLM provides auxiliary rewards by evaluating the semantic similarity between τ_t and $l_{m(t)}$, as illustrated in Figure 1. The use of VLM-based reward model transform the original MDP into a shaped MDP $\widetilde{\mathcal{M}} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, s_0, \tilde{r}, \gamma \rangle$, where the reward function becomes $\tilde{r} = r^e(s,a) + r^v(\tau_t, l_{m(t)})$.

4 Theoretical Analysis

In this work, we define "reward noise" as errors made by the VLM-based reward model when evaluating an agent's trajectory: either spuriously rewarding a trajectory that doesn't satisfy the instruction (false positive) or failing to reward one that does (false negative). In this section, we first show that, in the absence of noise, auxiliary rewards derived from language instructions, reflecting progress toward the goal, accelerate convergence compared to relying solely on sparse environmental rewards, r^e. We then prove that false positive rewards impede convergence more severely than false negatives under the Heuristic guided RL (HuRL) framework (Cheng et al., 2021).

4.1 Auxiliary Rewards and Convergence

With sparse rewards, the gradient landscape is nearly flat, making gradient-based updates indistinguishable from a random walk in parameter space (see Proposition N.1 in Appendix N). In this problem, we are interested in $D := \|\theta_{goal} - \theta_0\|$ which is the distance in parameter space from initial parameters θ_0 to goal parameters θ_{goal}. We make the following assumption:

Assumption 4.1. Expert knowledge can guide the parameter search along a path in parameter space, defined by a sequence of n intermediate parameter vectors $\theta_1, \ldots, \theta_n$, where each θ_i represents the parameters after learning subtask l_i. As a result, the overall distance D can be decomposed into segments: $D \approx \sum_{k=1}^{n-1} d_i$, where $d_i = \|\theta_{i+1} - \theta_i\|$.

We therefore prove the following proposition:

Proposition 4.2. *The sum of expected time for a series of random walks, each covering the shorter distance of an individual sub-task, is less than the expected time to travel the entire distance D in one long random walk:* $\frac{1}{n-1}\mathbb{E}[T_D] \leq \mathbb{E}\left[\sum_{i=1}^{n-1} T_{d_i}\right] < \mathbb{E}[T_D]$.

and thus show that subgoal-based auxiliary rewards improve the convergence of random walk optimization in a sparse reward landscape up to a factor of $(n-1)$. See Appendix D for the proof. The theoretical takeaway is that auxiliary rewards aligned with intermediate subgoals can accelerate convergence of RL algorithms compared to learning from a single delayed reward at the goal state. In practice, however, this improvement depends critically on the quality of the auxiliary reward signal — which we examine in the next section.

4.2 Connection to Heuristic-Guided RL

The reward noise issue can be framed within the context of HuRL by Cheng et al. (2021). HuRL mandates that auxiliary reward signals serve as heuristics, where $h : \mathcal{S} \to \mathbb{R}$ approximates the future total rewards an agent expects to get starting from state s under the optimal policy π^* (i.e., $h(s) \approx V^*(s)$). First of all, we establish the following assumption:

Assumption 4.3. The expert instruction sequence $L = l_1, l_2, \ldots, l_n$ specifies a sequence of state and action landmarks guiding the agent toward goal states $\mathcal{S}_G$. Landmarks are key states or key actions that must hold or occur in any trajectories that reach $\mathcal{S}_G$.

Assumption 4.3 is reasonable and reflects the quality of expert instructions, which are crucial for the agent to reach the goal efficiently.

We now present the important proposition connecting VLM-based reward models to HuRL:

Proposition 4.4. *In sparse reward settings, VLM-based reward models, as implemented in Algorithm 1, can be viewed as a heuristic function $h(s_t)$ in HuRL that estimates $V^*(s_t)$.*

The proof is in Appendix E. HuRL framework allows us to analyze how false positive rewards influence on *optimality gap*, defined as $V^*(s_0) - V^\pi(s_0)$, a metric that is used for theoretical analysis of the convergence speed of RL algorithms — the smaller the gap, the faster the convergence. We demonstrate that false positive rewards increase the upper bound of this gap, whereas false negative rewards maintain this upper bound.

4.3 False Positives and Heuristic Overestimation

We provide the formal definition of false positive and false negative rewards from both instruction-following (IF) and heuristic perspectives:

Definition 4.5 (False Positive Rewards)**.** A false positive reward occurs when:

IF Perspective: For a trajectory τ_t that does not satisfy instruction $l_{m(t)}$, the VLM-based reward $r^v(\tau_t, l_{m(t)}) > 0$.

Heuristic Perspective: The heuristic $h(s_t) > V^*(s_t)$, overestimating the optimal value of s_t.

Definition 4.6 (False Negative Rewards)**.** A false negative reward occurs when:

IF Perspective: The VLM-based reward $r^v(\tau_t, l_{m(t)}) \approx 0$ for a trajectory τ_t that *does* satisfy instruction $l_{m(t)}$.

Heuristic Perspective: The heuristic $h(s_t) \approx 0 < V^*(s_t)$, underestimating the optimal value of s_t.

Proposition 4.7. *False positive rewards in the IF perspective imply false positive rewards in the heuristic perspective.*

Proof. The heuristic is approximated as $h(s_t) \approx V^*(s_t) = 1 \cdot \gamma^{\widetilde{T}-t}$ in *sparse reward setting*, calculated based on an assumed optimal path length $\widetilde{T}$. Here, $\widetilde{T} - t$ measures the remaining steps towards the goal. When a trajectory τ_t receives a high reward but fails to fulfill instruction $l_{m(t)}$, it corresponds to a high $h(s_t)$, thus a low $\widetilde{T} - t$. Yet the agent must eventually reattempt $l_{m(t)}$ due to Assumption 4.3, s_t is actually further from the goal than estimated. Therefore, the actual distance $T - t$ to reach the goal will exceed $\widetilde{T} - t$, and $V^*(s_t)$, calculated with the actual T, will be smaller than $h(s_t)$, calculated with $\widetilde{T}$. This thus explains how a false

positive reward from an instruction-following (or VLM) perspective leads to an overestimation of the heuristic. □

Researchers have advocated the benefits of pessimistic value estimation to enhance the stability of RL algorithms (Kumar et al., 2020; Jin, Yang, and Wang, 2021). In HuRL, Cheng et al. (2021) further identify a beneficial property: when a heuristic is Bellman-consistent pessimism (i.e., $\max_a(\mathcal{B}h)(s,a) \geq h(s)$), it results in a capped upper bound on the optimality gap.

Definition 4.8 (Bellman-consistent Pessimistic h)**.** Recall Bellman equation of h: $(\mathcal{B}h)(s,a) = r(s,a) + \gamma\mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)}[h(s')]$. A heuristic function h is said to be *Bellman-consistent pessimism* with respect to an MDP $\mathcal{M}$ if $\max_a(\mathcal{B}h)(s,a) \geq h(s)$. This condition essentially means that the heuristic h never overestimates the true value of a state.

Proposition 4.9. *Even if the heuristic remains conservative for all successor states, a single* ***false positive*** *overestimation, i.e., $h(s) > V^*(s)$, can violate the pessimistic condition by causing $max_a(\mathcal{B}h)(s,a) < h(s)$.*

The proof is in Appendix F. This implies that maintaining a pessimistic heuristic is inherently fragile, as the presence of a false positive in any state disrupts the pessimistic condition.

Our result on the impact of false positive and false negative rewards on convergence is captured as follows:

Theorem 4.10. *Adapting the optimality gap analysis from Cheng et al. (2021), the gap $V^*(s_0) - V^\pi(s_0)$ in heuristic-guided RL decomposes into regret and bias terms. False negatives preserve the upper bound on bias, as they maintain the heuristic's pessimism, i.e., $h(s) \leq V^*(s)$. In contrast, false positives breaks the pessimistic property, strictly increase the bias without an upper bound, thereby leading to a larger optimality gap and slower convergence.*

See Appendix G for the formal version and detailed proof. In the next section, we present a case study on cosine similarity metrics, demonstrating how they contribute to false positive rewards in learned reward models.

5 False Positives From Cosine Similarity

This section identifies and discusses two fundamental issues with cosine similarity scores in sequential decision-making contexts: *state entanglement* and *composition insensitivity*. The former issue, state entanglement, refers to the metric's inability to recognize trajectories that, while being cosine similar to the target instruction in the embedding space, fail to reach the goal states in S_G. The latter issue refers to the metric's tendency to reward trajectories even when the temporal relationships between sub-tasks are not satisfied.

The Issue of State Entanglement State entanglement refers to the issue where the cosine similarity metric erroneously pays more attention to lexical-level similarity while lacking comprehension of the underlying state transitions. Consequently, rewards are given to trajectory-instruction pairs that are cosine similar in embedding space but in fact result in distinct state transitions. For instance, consider the

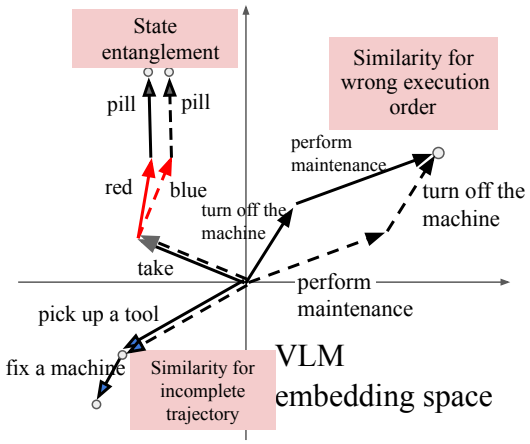


Figure 2: Schematic diagram of false positives in embedding space.

significant contrast between "take the *red* pill" and "take the *blue* pill". Despite their lexical similarity, they lead to different states. However, the cosine similarity metric may represent them as similar due to the shared words, disregarding the critical difference in state outcomes (see Figure 2 top left). Understanding state transitions is crucial in sequential decision-making scenarios. Otherwise, rewards may be given to trajectories that lead to unintended states, potentially prolonging the path to the goal state by necessitating corrective actions or re-attempts.

The Issue of Composition Insensitivity One might wonder why composition insensitivity persists despite using a pointer mechanism (Appendix C) that enforces sequential instruction order. The issue arises because sentences l in the instruction sequence L are not always *atomic*: a single instruction like "tidy the room before leaving" may implicitly encode multiple sub-tasks. While the pointer ensures progress through high-level steps, it does not help the VLM parse the internal structure of complex, non-atomic instructions. As a result, "composition insensitivity" emerges, misaligning the agent's trajectory and undermining reward accuracy. Unfortunately, **even state-of-the-art natural language understanding models struggle with decomposing atomic sentences without losing critical sequential or contextual information** (Dziri et al., 2024). Under this circumstance, composition insensitivity in cosine similarity metrics gives rise to two issues: (1) *rewarding incomplete task execution* – cosine similarity may incorrectly reward incomplete task execution by giving high scores even when critical steps are missing. We observed this phenomenon particularly in the *Montezuma* environment, where RL agents tend to *hack* the reward system by focusing on the easiest actions that yield rewards (e.g., moving towards a direction) rather than executing more complex, timely actions. **Eventually, this leads to an overestimation of the agent's progress towards the ultimate goal** (see Figure 2 bottom left). (2) *insensitivity to the ordering of execution* – VLM models often fails to adequately penalize incorrect execution sequences, rather, it assigns high rewards based merely

on the presence of relevant actions, disregarding their order (see Figure 2 top right). In contrast to some advancements in language models, compact visual and sentence embeddings from multimodal VLMs remain largely insensitive to sequential information (Pham et al., 2020). When the task is order-sensitive, executing actions in the wrong sequence prolongs the path towards the goal state, as agents need to re-attempt the correct order — yet the VLM's reward may still incentivize the agent to continue with the incorrect sequence. We empirically demonstrate these issues in Section 6, showing their substantial impact on agent learning in sparse reward environments.

6 Experiments on Reward Noise Impact

Our experiments test the following hypothesis: **(H1)** The two issues of *state entanglement* and *composition insensitivity* exist, **(H2)** *false positive* rewards are prevalent during training, **(H3)** VLM reward models lacking noise handling mechanisms underperform against intrinsic reward models in sparse reward environments, **(H4)** *false negatives* may not be as harmful as *false positives*.

Experimental Setup We evaluate these hypotheses through various challenging sparse-reward environments: (1) *Crafter*, an open-ended 2D Minecraft (Hafner, 2022); (2) *Montezuma*, a classic hard adventure game in Atari (Bellemare et al., 2013); and (3) *Minigrid 'Go To Seq'*, a hard task involving long-horizon navigation and object interactions (Chevalier-Boisvert et al., 2018). Details on why these environments are **preferred over other test benchmarks**, along with the evaluation metrics, are provided in Appendix A. A **Markovian** and a **Non-Markovian** reward model were tested: (1) *Pixl2R* by Goyal, Niekum, and Mooney (2020), which uses only the current video frame to determine if the goal state specified in the instruction has been reached; and (2) *ELLM-*, a variant of *ELLM* by Du et al. (2023) that directly uses preset expert instructions and compares them with the transition differences of the agent's trajectory. The VLM backbones used are: (1) *CLIP* (Radford et al., 2021), pretrained by image-text pairs; and (2) *X-CLIP* (Ma et al., 2022), pretrained by video-text pairs. To ensure high-quality finetuning data, we used internal information from the game engine to annotate expert trajectories from expert agents. To demonstrate how noisy reward signals hinder learning, we selected a strong intrinsic reward model *DEIR* (Wan et al., 2023) for comparison. It provides auxiliary rewards based on observation novelty to encourage exploration. The detailed VLM-based reward model procedure, finetuning process and training hyperparameters are in Appendix H.

Reward Noise Issue To investigate **H1** — the existence of two issues, *state entanglement* and *composition insensitivity*, in VLM-based reward models — we evaluated the models' sensitivity by examining how cosine similarity scores change for manipulated trajectory-instruction pairs. These manipulations, designed to probe to robustness against noise, included the following:

1. **Trajectory Reversal:** We inverted the sequence of frames within each trajectory (i.e., `frames = frame[::-1]`) to assess the model's ability to detect reversed state transitions. This manipulation tests whether the model can distinguish between forward and backward progression in the state transition.
2. **Instruction Negation:** We modified the original instructions by adding negation (e.g., changing "do l_k" to "do not do l_k" or "avoid l_k"). This tests the model's sensitivity to semantic changes in the instruction that fundamentally alter the goal.
3. **Concatenation and Order Swapping:** Given two trajectory-instruction pairs (τ_1, l_1) and (τ_2, l_2), we created concatenated pairs and then swapped the order in one modality. For example: a) Original concatenation: $(\tau_1+\tau_2, l_1+l_2)$; b) Swapped trajectory: $(\tau_2+\tau_1, l_1+l_2)$; c) Swapped instruction: $(\tau_1 + \tau_2, l_2 + l_1)$. This tests the model's sensitivity to the order of components in multi-step tasks.
4. **Concatenation with Partial Content:** We concatenated pairs but truncated one modality. For instance: a) Truncated trajectory: $(\tau_1, l_1 + l_2)$; b) Truncated instruction: $(\tau_1 + \tau_2, l_1)$. This assesses the model's ability to detect partial mismatches in longer sequences.

Our results reveal a critical flaw in the reward model: for manipulated pairs that fundamentally fail to fulfill the instruction, the model paradoxically assigns high similarity scores (see Figure 3 for overall results and also Appendix I for individual environments). Note that the poor performance in the negation case aligns with broader challenges in natural language processing. Recent studies (Hossain et al., 2022; Truong et al., 2023) have highlighted that negation is central to language understanding but is not properly captured by modern language models. This limitation extends to VLMs and directly leads to false positive rewards.

Prevalence of False Positives To address **H2**, we analyzed reward distribution heatmap from VLM-based reward models during training. The heatmap (Figure 5 left) revealed a concerning trend: RL agents engage in reward hacking, receiving rewards across vast areas of the environment rather than just at goal states. For instance, in *Montezuma* where the goal is to grab the key and escape the room, we observed that agents received rewards even for falling off cliffs, which undoubtedly contribute to false positive rewards. For environments without fixed camera views, we calculated the step offset between the current rewarded state and the actual goal state. A positive offset indicates a false positive reward, as the reward was given before reaching the goal. Conversely, a negative offset indicates a false negative, where the agent reached the goal but the reward model failed to acknowledge it (see Figure 5). Notably, besides positive offsets, we observed a large amount of negative offsets in Minigrid environments. We attribute this to Minigrid's abstract shape-based visual representations, which fall outside the VLM's training distribution.

Impact on Learning We trained agents using learned VLM reward models and compared

their learning efficacy against intrinsic reward models. As shown in Table 1, our results confirmed **H3**: *instruction-following RL agents using learned VLM reward models*

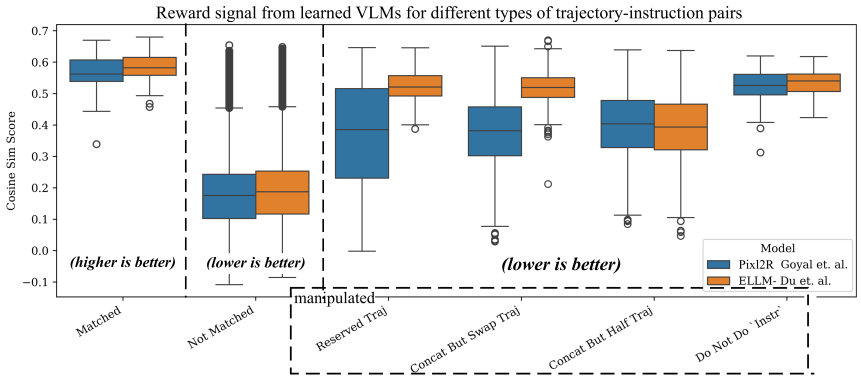


Figure 3: Learned VLM models performed badly with O.O.D. examples. They incorrectly assign high scores to manipulated pairs, which should be low as the trajectories in the manipulated pairs fail the instruction.

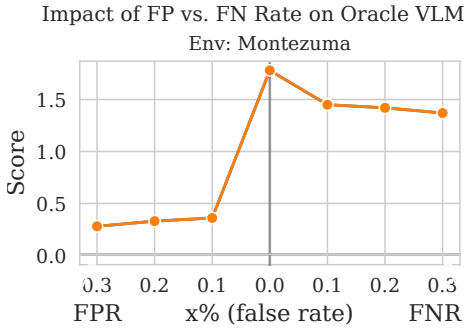


Figure 4: The false positive vs. false negative oracle model. The false positive model get a more severe drop in the final training score.

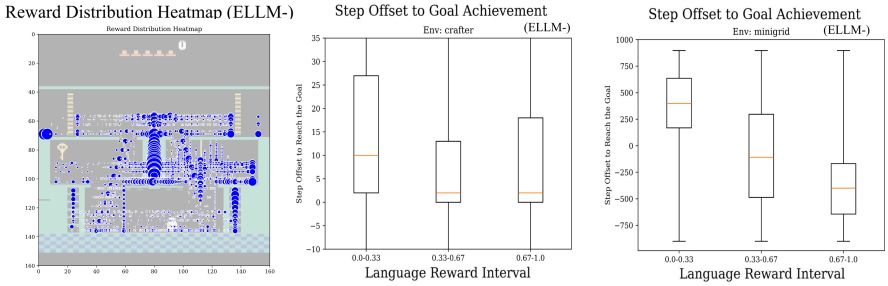


Figure 5: The heatmap shows rewards received at various locations, with larger circle sizes indicating higher rewards. The later figures shows the offsets between the state where rewards are given and the actual goal-reaching state. Agents are getting both issues of false positives and false negatives

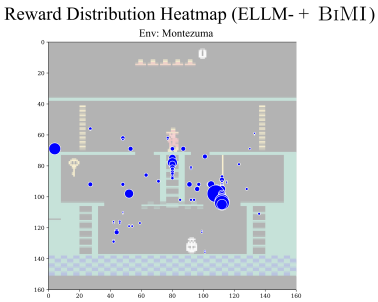


Figure 6: The ratio of false positive rewards is significantly reduced after applying BIMI

Table 1: Score metric across environments (equivalent to total rewards, higher is better). ⋆ denotes baseline intrinsic reward model. VLM reward models without noise handling underperformed. All are based on PPO.

Models	Type	Monte.	Minigrid	Crafter	% vs. DEIR
PPO	Pure	0.151	24.9	16.8	−28%
DEIR ⋆	Intrinsic	0.174	55.5	19.7	–
Pixl2R	VLM	0.142	12.4	9.40	**−49%**
ELLM-	VLM	0.150	19.4	10.8	−41%
Pixl2R + DEIR	VLM + intr.	0.176	17.3	10.4	−38%
ELLM- + DEIR	VLM + intr.	0.178	30.9	11.8	−27%

without noise handling consistently underperform compared to DEIR, the intrinsic reward-based RL agent.

In our efforts to assess the impact of false positive rewards from auxiliary reward model without the interference of other factors such as domain shift, poor data quality, and errors from other issues such as the choice of multimodal architectures, we devised a **simulated** reward model that access to internal state information from the game engine. Experiments with these simulated models provide initial evidence of the differential impact of false negatives versus false positives on training outcomes, as posited by **H4**, suggesting that false positives — particularly those tied to temporal insensitivity — more severely degrade final scores. Due to page limits, we move the details to Appendix J.

To compare the effects of reward noise, we designed an oracle Pixl2R model with two variants: one simulating false negatives by randomly removing $x\%$ of true subgoal rewards, and one simulating false positives by adding small one-off rewards (0.1) to $x\%$ of irrelevant states. This design reflects a realistic failure mode of VLMs, where false positive rewards — despite being only 1/10 the magnitude of true subgoal rewards and sometimes located far from the optimal trajectory — can still be exploited by the agent through training iterations, a phenomenon known as *reward hacking*. In contrast, false negatives merely reduce the frequency of positive feedback without misleading the agent. The results indicate that false negatives were less detrimental to agent performance than false positives (see Figure 4). This aligns with our theoretical analysis in Theorem 4.10.

7 Addressing the False Positive Issue

Binary Signal and Conformal Prediction Having established the detrimental effects of false positive rewards, our solution is to strategically manage the trade-off: reducing false positives, even at the cost of a slight increase in false negatives. First, we propose a reward function that issues

a one-time binary reward only when the similarity between the agent's current trajectory and the instruction exceeds a high confidence threshold. This approach contrasts with previous methods, which provide continuous rewards whenever the reward score exceeds a predefined threshold, and continue to do so until reaching a maximum cap. Our method, however, delivers this reward only once. This approach minimizes the likelihood of accumulating false positive rewards while maintaining adherence to Proposition 4.4.

To achieve this, we introduce a thresholding mechanism using a calibration set of true positive trajectory-instruction pairs. This threshold, denoted as $\hat{q}$, is set to the empirical quantile of cosine similarity scores at the significance level $1-\alpha$. Pairs whose similarity scores fall below this threshold $\hat{q}$ receive no reward. Conversely, pairs exceeding $\hat{q}$ receive a one-time $+1$ reward:

$$r^v_{\text{BI}}(\tau, l_k) = \mathbf{1}_{\{p(l_k|\tau)\geq\hat{q}\}} \tag{1}$$

It statistically guarantees a high probability (at least $1-\alpha$) that the true positive pairs are recognized (i.e., above the threshold) while minimizing the average number of mistakes predicting false positives (Sadinle et al., 2019). The threshold calculation is detailed in Algorithm 2.

Mutual Information Maximization Intuitively, when we observe rewards coming from a particular signal source too frequently, we tend to downplay the significance of that signal to avoid over-reliance. This intuition is effectively captured by incorporating a *mutual information maximization* term into the reward function. Specifically, the updated reward function $r^v_{\text{MI}}(\tau, l_k)$ measures the mutual information between the agent's trajectory and the instruction. Formally, it can be expressed as:

$$\begin{aligned} r^v_{\text{MI}}(\tau, l_k) &= I(l_k;\tau) = D_{KL}(p(l_k,\tau) \,||\, p(l_k)p(\tau)) \\ &= \mathbb{E}_{\tau\sim\pi_\theta, l_k\sim L}[\log p(l_k \mid \tau) - \log p(l_k)] \end{aligned} \tag{2}$$

where $\tau = \langle s_{t-W}, a_{t-W}, \ldots, s_t\rangle$ is the agent's trajectory up to current time step t, and W is the memory size of the agent for its past trajectory. $p(l_k \mid \tau)$ comes from the similarity score provided by VLMs, referring to the likelihood of the instruction l_k being fulfilled by trajectory τ. $p(l_k)$ is overall likelihood of encountering the instruction l_k in the learning environment. Therefore, the second term in the equation serves as a regularization term that downplays the significance of the reward signal when it is too frequent. For instance, if a VLM frequently detects that the agent's actions are fulfilling the "climbing the ladder" instruction, even when the agent is performing unrelated tasks, any reward signal from this instruction will be downplayed.

We treat the set of instructions $\{l_k\}$ as a finite discrete support and define an empirical frequency:

$$s(l_k) \;=\; \mathbb{E}_{\tau\sim\pi_{\theta_{-1}}}\Big[\tfrac{1}{T_\tau}\sum_{t=1}^{T_\tau}\mathbf{1}\big\{p(l_k \mid \tau_t)\geq\hat{q}\big\}\Big], \tag{3}$$

where τ_t is the agent's trajectory up to time t, and we count one whenever the VLM scores τ_t as fulfilling l_k over the total trajectory length T_τ. We then normalize to obtain a *probability mass function* (PMF) over instructions: $p(l_k) = \frac{s(l_k)}{\sum_j s(l_j)}$, and $\sum_j s(l_j) = 1$. Because $\{l_k\}$ is discrete, $p(l_k)$ is a valid PMF (not a continuous density) and can be plugged directly into the usual discrete mutual information in eq. (2). The subscript θ_{-1} in $\pi_{\theta_{-1}}$ indicates that the trajectories are sourced from rollouts in the previous policy iteration, acknowledging the impracticality of real-time computation of $p(l_k)$ during an ongoing episode.

To enhance the stability of the training process, we adopt a linearized version of the mutual information maximization approach, as proposed by Li et al. (2023). Overall, BIMI, the proposed reward function that enhances the noise resilience of VLM-based reward models, can be expressed as follows:

$$r^v_{\text{BIMI}}(\tau, l_k) = \max(\mathbf{1}_{\{p(l_k|\tau)\geq\hat{q}\}} - p(l_k), 0) \tag{4}$$

Note that the BIMI approach primarily mitigates false positives (FPs) rather than false negatives (FN). Both BI and MI aim to reduce the likelihood of rewarding unintended trajectories, thus addressing the FP issue. While this conservative approach may increase false negatives by pruning some correct trajectories, this trade-off is beneficial, as demonstrated by both our empirical and theoretical results.

8 Experiments and Results

We follow the same experimental setup as in Section 6 to ensure a fair comparison. Furthermore, we set confidence level for empirical quantile calculation to be $1-\alpha = 0.9$. We adhered to the standard requirement of limiting the training budget to 1 million frames (Hafner, 2022).

Montezuma Pixl2R+BIMI demonstrated 14% performance increase compared to the original models (see Table 2), which is slightly below our expectations. We attribute this result to BIMI's intentional strategy of providing less frequent discrete rewards. While this strategy effectively reduces FPs, it does not substantially mitigate the inherent reward sparsity issue in *Montezuma*. However, **we discovered a remarkable synergy between BIMI and intrinsic reward models.** While previous models showed no significant improvements with *DEIR* (the intrinsic reward model) alone, combining BIMI and *DEIR* led to a 65% performance gain. The gap in collaboration effectiveness can be attributed to two factors. In the previous setup, the consistent presence of false positive rewards misled agents towards unacceptable behaviors and hindered further exploration. Now, BiMI's less frequent but more meaningful rewards provide anchor points for the agent's learning. Meanwhile, *DEIR*'s intrinsic rewards fill the gaps between these anchor points, encouraging the agent to explore efficiently in the interim.

See Figure 5 left and Figure 6 for a **quantitative analysis**: BIMI rewards are now concentrated on key locations. A significant improvement is the minimal rewards given for falling off cliffs, which was a common source of FPs in the original model. Figure 7 demonstrates a higher success rate in grabbing the key in the first room, one of the most difficult tasks in *Montezuma*, highlighting the effectiveness of the proposed reward function and its synergy with intrinsic reward models in guiding agents to solve difficult sparse-reward tasks.

Table 2: Model score across various environments. ⋆ is the baseline agents with a learned VLM-based reward model to compare with. BIMI significantly improves performance in *Montezuma* and *Minigrid*, while showing mixed results in *Crafter*

Methods	Montezuma	% vs. ⋆	Minigrid	% vs. ⋆	Crafter	% vs. ⋆
Pixl2R ⋆	0.142 ± 0.003	–	12.4 ± 2.43	–	9.40 ± 0.022	–
Pixl2R + Bi	0.137 ± 0.009	−3.5%	31.2 ± 2.04	+151%	10.7 ± 0.784	+14%
Pixl2R + BiMI	0.162 ± 0.022	+14%	37.5 ± 7.83	+199%	7.95 ± 0.351	−15%
Pixl2R + DEIR	0.176 ± 0.009	+23%	17.3 ± 0.51	+39%	10.4 ± 1.015	+10%
Pixl2R + BiMI + DEIR	0.267 ± 0.016	+88%	57.7 ± 2.15	+364%	11.0 ± 0.190	+17%
ELLM- ⋆	0.150 ± 0.004	–	19.4 ± 10.06	–	10.8 ± 1.017	–
ELLM- + Bi	0.151 ± 0.016	+0.6%	29.7 ± 1.29	+53%	11.1 ± 0.601	+3.2%
ELLM- + BiMI	0.156 ± 0.014	+4.0%	33.6 ± 3.99	+74%	9.42 ± 0.267	−12%
ELLM + DEIR	0.178 ± 0.029	+20%	30.9 ± 3.50	+59%	11.8 ± 1.152	+9.5%
ELLM- + BiMI + DEIR	0.279 ± 0.078	+86%	56.2 ± 6.19	+190%	13.1 ± 0.393	+22%

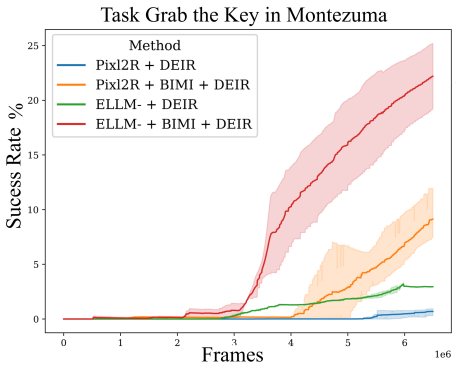


Figure 7: BIMI reward showed faster and higher success rates on difficult tasks in Montezuma

Minigrid: ELLM-+BIMI achieved a remarkable 74% improvement in performance compared to the original models.This substantial gain is particularly noteworthy given the unique challenges presented by *Minigrid*. The abstract, shape-based visuals diverge drastically from the natural images used in VLMs' pretraining, preventing the models from effectively utilizing their prior pretraining knowledge. Consequently, VLMs struggled to accurately assess similarities between Minigrid's abstract visuals and textual instructions, resulting in highly noisy reward signals. The significant improvement demonstrated by BIMI underscores its effectiveness in handling noisy signals, directly addressing our primary research challenge. This capability is crucial for deploying instruction-following agents in real-world, unfamiliar scenarios, where visual inputs often deviate from the VLMs' training distribution, leading to noisy reward signals.

Crafter: We observed an intriguing pattern of results. The BI component alone led to 14% and 3.2% improvement in performance over the original models. However, contrary to our observations in other environments, the addition of the MI component actually decreased this improvement. This unexpected outcome can be attributed to the unique nature of *Crafter* task, where agents must repeatedly achieve the same subtasks (e.g., drinking water) for survival. The MI component, designed to discourage over-reliance on frequently occurring signals, inadvertently penalized the necessary repetition of survival-critical actions. Nevertheless, BI alone still managed to improve performance over vanilla VLM-based reward models, suggesting that reducing FPs is still beneficial across all testing environments. The combination of BIMI with *DEIR* (the intrinsic reward model) also showed promising results, indicating a productive balance between exploration (driven by *DEIR*) and exploitation (guided by BIMI instruction reward).

Overall Performance and Ablation Study The BIMI reward function yielded significant performance gains, as detailed in Table 2. For the Markovian *Pixl2R* model, BIMI improved scores by 67%, while the non-Markovian *ELLM-* model improved by 22%. These improvements, alongside synergy with intrinsic rewards, are depicted in Figure 16, which illustrates how combining BIMI with intrinsic rewards (i.e., DEIR) boosts agent performance across different environments.

Our ablation study further demonstrates the distinct contributions of the binary reward (BI) and mutual information (MI) components within the BIMI framework with results shown in Figure 17. The BI mechanism alone drove a 36.5% performance increase over baseline models. Excluding the *Crafter* environment, the MI component added a further 23% improvement over BI alone. Together, these results underscore the importance of both components in mitigating false positive rewards and enhancing agent performance.

9 Conclusion and Discussion

Most existing work on VLM-based reward models for embodied instruction-following agents remains at the proof-of-concept stage in simplified environments (Du et al., 2023; Rocamonde et al., 2024; Wang et al., 2024), overlooking the challenge of noisy rewards in complex, long-horizon tasks. Our work addresses this gap through theoretical analysis grounded in *heuristic admissibility* — a classical concept from the automated planning domain but rarely applied to embodied learning agents. This perspective enables us to rigorously explain why VLM-based reward models are prone to failure without explicit noise handling. Building on this foundation, we identify two key insights: (1) false positive rewards are substantially more harmful to policy learning than false negatives, and (2) the proposed BIMI reward function, which incorporates pessimistic rewarding, effectively mitigates this problem. Our findings are supported across three challenging embodied tasks, spanning Markovian and non-Markovian reward models.

Limitations We primarily focused on linear sequences of language instructions, excluding more complex cases. Future research should investigate conditional and ambiguous instructions, which likely introduce additional challenges for VLM-based reward models. There is also a gap in providing a rigorous theoretical foundation for why our theoretical findings extend to non-Markovian reward models. However, with advancements in deep RL, the distinction between non-Markovian and Markovian models has become increasingly blurred (Hausknecht and Stone, 2015).

References

Agarwal, A.; Kakade, S. M.; Lee, J.; and Mahajan, G. 2019. On the Theory of Policy Gradient Methods: Optimality, Approximation, and Distribution Shift. *J. Mach. Learn. Res.*, 22: 98:1–98:76.

Bai, C.; et al. 2024. Pessimistic value iteration for multi-task data sharing in Offline Reinforcement Learning. *Artificial Intelligence*, 326: 104048.

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2013. The Arcade Learning Environment: An Evaluation Platform for General Agents. *J. Artif. Intell. Res.*, 47: 253–279.

Brafman, R. I.; and Tennenholtz, M. 2002. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct): 213–231.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI Gym. arXiv:1606.01540.

Burda, Y.; Edwards, H.; Storkey, A.; and Klimov, O. 2018. Exploration by random network distillation. In *International Conference on Learning Representations*.

Chan, H.; Mnih, V.; Behbahani, F.; Laskin, M.; Wang, L.; Pardo, F.; Gazeau, M.; Sahni, H.; Horgan, D.; Baumli, K.; Schroecker, Y.; Spencer, S.; Steigerwald, R.; Quan, J.; Comanici, G.; Flennerhag, S.; Neitz, A.; Zhang, L. M.; Schaul, T.; Singh, S.; Lyle, C.; Rocktäschel, T.; Parker-Holder, J.; and Holsheimer, K. 2023. Vision-Language Models as a Source of Rewards. In *Second Agent Learning in Open-Endedness Workshop*.

Chen, Y.-C.; Kochenderfer, M. J.; and Spaan, M. T. 2018. Improving offline value-function approximations for POMDPs by reducing discount factors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3531–3536. IEEE.

Cheng, C.; et al. 2021. Heuristic-Guided Reinforcement Learning. In *NeurIPS*, 13550–13563.

Chevalier-Boisvert, M.; Bahdanau, D.; Lahlou, S.; Willems, L.; Saharia, C.; Nguyen, T. H.; and Bengio, Y. 2018. BabyAI: A Platform to Study the Sample Efficiency of Grounded Language Learning. In *International Conference on Learning Representations*.

Chevalier-Boisvert, M.; Dai, B.; Towers, M.; Perez-Vicente, R.; Willems, L.; Lahlou, S.; Pal, S.; Castro, P. S.; and Terry, J. K. 2023. Minigrid & Miniworld: Modular & Customizable Reinforcement Learning Environments for Goal-Oriented Tasks. In *NeurIPS*.

Corazza, J.; Gavran, I.; and Neider, D. 2022. Reinforcement Learning with Stochastic Reward Machines. In *AAAI*, 6429–6436. AAAI Press.

Du, Y.; Watkins, O.; Wang, Z.; Colas, C.; Darrell, T.; Abbeel, P.; Gupta, A.; and Andreas, J. 2023. Guiding Pretraining in Reinforcement Learning with Large Language Models. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, 8657–8677. PMLR.

Dziri, N.; et al. 2024. Faith and fate: Limits of transformers on compositionality. *Advances in Neural Information Processing Systems*, 36.

Frome, A.; Corrado, G. S.; Shlens, J.; Bengio, S.; Dean, J.; Ranzato, M.; and Mikolov, T. 2013. DeViSE: A Deep Visual-Semantic Embedding Model. In *NIPS*, 2121–2129.

Fu, Y.; Zhang, H.; Wu, D.; Xu, W.; and Boulet, B. 2024. FuRL: Visual-Language Models as Fuzzy Rewards for Reinforcement Learning. In *ICML*. OpenReview.net.

Ghosal, G. R.; Zurek, M.; Brown, D. S.; and Dragan, A. D. 2022. The Effect of Modeling Human Rationality Level on Learning Rewards from Multiple Feedback Types. In *AAAI Conference on Artificial Intelligence*.

Glorot, X.; and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, volume 9 of *JMLR Proceedings*, 249–256. JMLR.org.

Goyal, P.; Niekum, S.; and Mooney, R. J. 2019. Using Natural Language for Reward Shaping in Reinforcement Learning. In *IJCAI*, 2385–2391. ijcai.org.

Goyal, P.; Niekum, S.; and Mooney, R. J. 2020. PixL2R: Guiding Reinforcement Learning Using Natural Language by Mapping Pixels to Rewards. In *CoRL*, volume 155 of *Proceedings of Machine Learning Research*, 485–497. PMLR.

Hafner, D. 2022. Benchmarking the Spectrum of Agent Capabilities. In *ICLR*. OpenReview.net.

Hausknecht, M. J.; and Stone, P. 2015. Deep Recurrent Q-Learning for Partially Observable MDPs. In *AAAI Fall Symposia*, 29–37. AAAI Press.

Hossain, M. M.; et al. 2022. An analysis of negation in natural language understanding corpora. *arXiv preprint arXiv:2203.08929*.

Icarte, R. T.; Klassen, T. Q.; Valenzano, R. A.; and McIlraith, S. A. 2018. Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, 2112–2121. PMLR.

Jin, Y.; Yang, Z.; and Wang, Z. 2021. Is pessimism provably efficient for offline rl? In *International Conference on Machine Learning*, 5084–5096. PMLR.

Kaplan, R.; et al. 2017. Beating Atari with Natural Language Guided Reinforcement Learning. *ArXiv*, abs/1704.05539.

Kumar, A.; Zhou, A.; Tucker, G.; and Levine, S. 2020. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191.

Li, M.; Zhao, X.; Lee, J. H.; Weber, C.; and Wermter, S. 2023. Internally rewarded reinforcement learning. In *International Conference on Machine Learning*, 20556–20574. PMLR.

Lin, X.; Wang, Y.; Olkin, J.; and Held, D. 2020. SoftGym: Benchmarking Deep Reinforcement Learning for Deformable Object Manipulation. In *Conference on Robot Learning*.

Liu, H.; Li, C.; Wu, Q.; and Lee, Y. J. 2023. Visual Instruction Tuning. In *NeurIPS*.

Ma, Y.; Xu, G.; Sun, X.; Yan, M.; Zhang, J.; and Ji, R. 2022. X-CLIP: End-to-End Multi-grained Contrastive Learning for Video-Text Retrieval. *Proceedings of the 30th ACM International Conference on Multimedia*.

Mahmoudieh, P.; Pathak, D.; and Darrell, T. 2022. Zero-Shot Reward Specification via Grounded Natural Language. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, 14743–14752. PMLR.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M. A.; Fidjeland, A.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nat.*, 518(7540): 529–533.

Moon, S.; Yeom, J.; Park, B.; and Song, H. O. 2023. Discovering Hierarchical Achievements in Reinforcement Learning via Contrastive Learning. In *Neural Information Processing Systems*.

Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In *ICML*, 278–287. Morgan Kaufmann.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C. L.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; Schulman, J.; Hilton, J.; Kelton, F.; Miller, L.; Simens, M.; Askell, A.; Welinder, P.; Christiano, P. F.; Leike, J.; and Lowe, R. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*.

Pham, T. M.; Bui, T.; Mai, L.; and Nguyen, A. 2020. Out of Order: How important is the sequential order of words in a sentence in Natural Language Understanding tasks? *arXiv preprint arXiv:2012.15180*.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; Krueger, G.; and Sutskever, I. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, 8748–8763. PMLR.

Rocamonde, J.; Montesinos, V.; Nava, E.; Perez, E.; and Lindner, D. 2024. Vision-Language Models are Zero-Shot Reward Models for Reinforcement Learning. In *ICLR*. OpenReview.net.

Sadinle, M.; et al. 2019. Least ambiguous set-valued classifiers with bounded error levels. *Journal of the American Statistical Association*, 114(525): 223–234.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347.

Shridhar, M.; et al. 2022. Cliport: What and where pathways for robotic manipulation. In *Conference on robot learning*, 894–906. PMLR.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, 5026–5033. IEEE.

Truong, T. H.; Baldwin, T.; Verspoor, K.; and Cohn, T. 2023. Language models are not naysayers: an analysis of language models on negation benchmarks. *arXiv preprint arXiv:2306.08189*.

Tschannen, M.; Gritsenko, A.; Wang, X.; Naeem, M. F.; Alabdulmohsin, I.; Parthasarathy, N.; Evans, T.; Beyer, L.; Xia, Y.; Mustafa, B.; Hénaff, O.; Harmsen, J.; Steiner, A.; and Zhai, X. 2025. SigLIP 2: Multilingual Vision-Language Encoders with Improved Semantic Understanding, Localization, and Dense Features. arXiv:2502.14786.

Uehara, M.; and Sun, W. 2021. Pessimistic model-based offline reinforcement learning under partial coverage. *arXiv preprint arXiv:2107.06226*.

Wan, S.; Tang, Y.; Tian, Y.; and Kaneko, T. 2023. DEIR: Efficient and Robust Exploration through Discriminative-Model-Based Episodic Intrinsic Rewards. In *International Joint Conference on Artificial Intelligence*.

Wang, X.; Huang, Q.; Celikyilmaz, A.; Gao, J.; Shen, D.; Wang, Y.; Wang, W. Y.; and Zhang, L. 2019. Reinforced Cross-Modal Matching and Self-Supervised Imitation Learning for Vision-Language Navigation. In *CVPR*, 6629–6638. Computer Vision Foundation / IEEE.

Wang, Y.; Sun, Z.; Zhang, J.; Xian, Z.; Biyik, E.; Held, D.; and Erickson, Z. 2024. RL-VLM-F: Reinforcement Learning from Vision Language Foundation Model Feedback. In *ICML*. OpenReview.net.

Wang, Z.; Cai, S.; Chen, G.; Liu, A.; Ma, X.; and Liang, Y. 2023. Describe, Explain, Plan and Select: Interactive Planning with LLMs Enables Open-World Multi-Task Agents. In *NeurIPS*.

Yu, T.; Quillen, D.; He, Z.; Julian, R.; Hausman, K.; Finn, C.; and Levine, S. 2019. Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning. In *CoRL*, volume 100 of *Proceedings of Machine Learning Research*, 1094–1100. PMLR.

Yuan, H.; Li, X.; Zhang, T.; Huang, Z.; Xu, S.; Ji, S.; Tong, Y.; Qi, L.; Feng, J.; and Yang, M.-H. 2025. Sa2VA: Marrying SAM2 with LLaVA for Dense Grounded Understanding of Images and Videos. arXiv:2501.04001.

Zhang, T.; Xu, H.; Wang, X.; Wu, Y.; Keutzer, K.; Gonzalez, J. E.; and Tian, Y. 2021. NovelD: A Simple yet Effective Exploration Criterion. In *Neural Information Processing Systems*.

Technical Appendix

Appendix Table of Contents

A Anticipatory defense on the choice of test environments and metrics

As discussed in Appendix B, we need test environments that clearly demonstrate the negative impact of noisy reward signals from VLM-based reward models. This requires complex state spaces (both large and diverse) where the approximation errors of VLMs become evident — specifically, where they struggle to provide accurate alignment scores between an agent's trajectory and given instructions due to the complexity of visual observations. Additionally, we need longer-horizon tasks where the impact of noisy reward signals can accumulate over time. In contrast, many embodied environments used in previous studies are not suitable due to their short-horizon tasks and simple state spaces. Table 3 lists common RL embodied environments used in the literature, explaining why certain environments are inadequate for our evaluation purposes while highlighting suitable environments we aim to use.

Table 3: Common RL embodied environments and their suitability for evaluating noisy reward signals from VLM-based reward models.

Env	Type	State Space Complexity	Horizon Length	Instruction Source
Not Suitable Envs				
CartPole (Brockman et al., 2016)	Classic Control	Simple	Short	Manual annotation
MountainCar (Brockman et al., 2016)	Classic Control	Simple	Short	Manual annotation
Humanoid (Todorov, Erez, and Tassa, 2012)	Bipedal robot control	Complex	Short	Manual annotation
MetaWorld (Yu et al., 2019)	Robotic arm	Moderate	Short	Predefined task descriptions
SoftGym (Lin et al., 2020)	Robotic arm	Moderate	Short	Predefined task descriptions
Suitable Envs				
Montezuma (Bellemare et al., 2013)	Platformer	Complex	Long	Manual annotation
Crafter (Hafner, 2022)	Open world survival	Complex	Long	Programmable (via engine data)
Minigrid (Chevalier-Boisvert et al., 2023)	Grid-world maze	Controllable	Controllable	Predefined task descriptions

We describe each testing environment used in our experiments. More details introduction can be found in on the official project homepage of each benchmark (Hafner, 2022; Bellemare et al., 2013; Chevalier-Boisvert et al., 2023).

- **Crafter** features randomly generated 2D worlds where the player needs to forage for food and water, find shelter to sleep, defend against monsters, collect materials, and build tools. The original Crafter environment does not have a clear goal trajectory or instructions; agents are aimed at surviving as long as possible and exploring the environment to unlock new crafting recipes. We modified the environment to include a preset linear sequence of instructions to guide the agent to mine diamond. However, this instruction was found to hinder the agent's performance. The nature of the task requires dynamic strategies and real-time decision-making, but the fixed instructions limited the agent. For example, the instruction did not account for what to do when the agent is attacked by zombies.

- **Montezuma's Revenge** is a classic adventure platform game where the player must navigate through a series of rooms to collect treasures and keys. The game is known for its sparse rewards and challenging exploration requirements. We manually annotate 97 instructions for the agent to follow, guiding it to conquer the game. The instructions were designed to guide the agent through the game's key challenges, such as avoiding enemies, collecting keys, and unlocking doors.

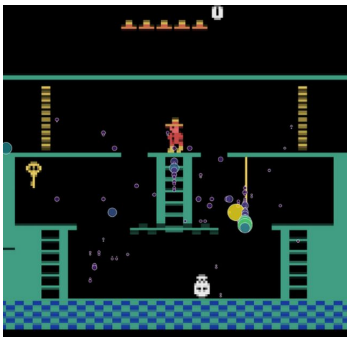
- **Minigrid 'Go to seq' Task**: We use the 'Go to seq' task in the Minigrid environment, where the agent must navigate through a sequence of rooms and touch target objects in the correct order. This is a sparse reward task where the agent receives a reward of 1 only upon completing the entire sequence correctly. During the training phase, we randomly generate 50 different tasks, each with a room size of 5, 3 rows, and 3 columns. Each task features a unique room layout and target object sequence. The instruction complexity is set to 3, meaning there are at least 3 target objects to interact with in a specific order.

Crafter and Instructions

1	Goal (Collect diamond)	
2	1. Collect wood	2. Collect wood again
3	3. Place table	4. Make wood pickaxe
4	5. Collect stone	6. Make stone pickaxe
5	7. Collect stone	8. Place furnace
6	9. Collect coal	10. Collect iron
7	11. Make iron pickaxe	12. Collect diamond

Figure 8: Illustration of the Crafter task. The agent must survive as long as possible and explore for new crafting recipes.

Montezuma and Instructions



	Goal	Room	x	y
2	climb down the middle ladder	1	74	192
3	jump right to the yellow rope	1	109	199
4	jump right to the right platform	1	133	192
5	climb down the right ladder	1	133	148
6	jump over the skull to its left	1	76	151
7	climb up the left ladder	1	21	192
8	jump up to grab the key	1	13	209
9	jump left to the left roof	1	44	235

Figure 9: Illustration of the Montezuma's Revenge task. The agent must navigate through a series of rooms to collect treasures and keys.

Minigrid and Instructions

	Goal
2	go to the grey door after you go to the blue door and go to the purple door
3	go to the box, then go to the yellow ball and go to a red door
4	go to the purple door and go to the purple door, then go to the red box
5	go to the green box, then go to a box
6	go to the ball and go to the yellow door after you go to a grey door
7	go to a ball, then go to the blue ball

Figure 10: Illustration of the Minigrid 'Go to seq' task. The agent must navigate through a sequence of rooms and touch target objects in the correct order.

A.1 Metrics

In our experiments, we used a score metric adapted from the Crafter benchmark (Hafner, 2022) to evaluate agent performance across different environments. This score metric aggregates success rates for individual subtasks using a geometric mean.

Formally, the score metric is defined as follows:

$$\text{Score} = \exp\left(\frac{1}{N}\sum_{k=1}^{N}\ln(1+s_k)\right) - 1 \tag{5}$$

where s_k is the agent's success rate of achieving instruction l_k, and N is the total number of instructions.

This metric was chosen over the *maximum total rewards* metric for several reasons:

1. **Consistency in Sparse Reward Settings:** Sparse reward environments often pose significant challenges for reinforcement learning agents. An agent might occasionally achieve high rewards by chance in one rollout but fail to replicate this success consistently in subsequent rollouts. This variability can lead to misleading evaluations if only the maximum total rewards are considered. The Score metric, by measuring the success rate of achieving each subgoal, provides a more stable and consistent measure of an agent's performance.
2. **Capturing Learning Stability:** The Score metric evaluates the agent's ability to consistently reproduce successful behaviors across multiple episodes. This is crucial in sparse reward settings, where the agent's performance can fluctuate significantly. By focusing on the success rates of individual subtasks, the Score metric offers a more granular and reliable assessment of the agent's learning progress and stability.
3. **Crafter Benchmark Standard:** The Crafter benchmark, which introduces the Score metric, is a well-regarded standard.

Crafter codebase provides *score* metric calculation by default. For Minigrid and Montezuma environments, we use the internal information from the game engine to detect whether the subtasks are completed, thus facilitating the calculation of the *score* metric.

To evaluate the **learning speed** of our RL algorithms, we utilize the Area Under the Learning Curve (AUC) metric. This metric, previously employed in the literature (Goyal, Niekum, and Mooney, 2019, 2020), quantifies the cumulative performance of the agent throughout the training process, effectively indicating how rapidly the agent improves its policy. We implement a win cap, limiting the maximum number of wins an agent can achieve during training to a certain number (e.g., 5000). Training automatically terminates once this limit is reached, under the assumption that policy learning has **stabilized** by this point. The AUC is therefore number of wins normalized by the total number of training episodes. Formally, this is expressed as:

$$\text{AUC} = \frac{\sum_{i=0}^{T}\text{wins}_i}{T}$$

where wins_i is the number of wins at episode i, and T is the total number of training episodes.

B Detailed Related Work

The approach of converting natural language instructions into reward signals for RL agents has been a longstanding area of exploration. Early studies (Kaplan et al., 2017; Goyal, Niekum, and Mooney, 2019; Wang et al., 2019; Goyal, Niekum, and Mooney, 2020) trained separate text and vision encoders from scratch so as to convert multimodal data into continuous feature vectors, then used a discriminator to assess alignment between trajectories and instructions. The cosine similarity metric, found by Frome et al. (2013) as an effective measure of semantic alignment, became foundational to these methods. Strikingly, the core architecture of VLM-based reward models, as illustrated in Figure 1, has remained largely unchanged throughout the years. Both pioneering efforts (Kaplan et al., 2017; Goyal, Niekum, and Mooney, 2019; Wang et al., 2019; Goyal, Niekum, and Mooney, 2020) and modern implementations continue relying on computing cosine similarity scores between semantic embeddings of instructions and agent trajectories to generate rewards. Formally, given two embedding vectors, $E(l)$ and $E(\tau)$, which denote the semantic embeddings of an instruction l and a trajectory τ, respectively, the cosine similarity is defined as:

$$\text{Cosine Similarity}(E(l), E(\tau)) = \frac{E(l)\cdot E(\tau)}{\|E(l)\|\|E(\tau)\|} \tag{6}$$

However, before the advent of CLIP (Radford et al., 2021) in 2021, these efforts were hampered by the absence of powerful pre-trained encoder-based VLMs. Consequently, researchers focused on proof-of-concept studies, testing their approaches on simpler, short-horizon tasks or sub-tasks — for example, Goyal, Niekum, and Mooney (2019) evaluated their method on sub-tasks rather than the full Montezuma's Revenge game[3]. This likely reflected an understanding that complex, long-horizon tasks were impractical at the time. As such, the field had yet to grapple with noisy reward signals, which remained minimal in these simpler environments.

CLIP (Radford et al., 2021), an encoder-based VLM, marked a turning point by providing a high-quality joint semantic embeddings for vision and language data. Leveraging CLIP, researchers started addressing more complex environments. For example, Mahmoudieh, Pathak, and Darrell (2022) used a CLIP-based reward model to guide a robotic arm in manipulation tasks, demonstrating its feasibility for real-world applications.

[3]available at https://www.retrogames.cz/play_124-Atari2600.php

In addition to the topics mentioned in the main body of the paper, we continue to highlight related work on reward signals and mitigation strategies below.

Reward Signal from Human Preference. Recent work on RL from Human Feedback (RLHF) (Ouyang et al., 2022) also leverage expert preference as a reward signal. However, our work differs in key aspects. Unlike RLHF's focus on textual outputs, our approach involves evaluating cross-modal similarities between visual and textual data in environments requiring long-horizon decision-making and frequent embodied interactions, a domain not typically covered by RLHF.

Mitigating Misspecified Rewards. Prior works proposed mitigating false positive rewards by training a parallel exploration policy to escape local optima caused by misspecified rewards (Ghosal et al., 2022; Fu et al., 2024). In contrast, we propose a novel reward function that directly penalizes likely false positive reward signals during training. We further show that our method complements exploration-based methods and achieves superior performance when combined.

Research Gap Successes with CLIP-based reward models have been particularly pronounced in environments with **short-horizon** tasks. For instance, Rocamonde et al. (2024) achieved promising results in classic control tasks such as CartPole, MountainCar (Brockman et al., 2016), and Humanoid (Todorov, Erez, and Tassa, 2012), environments that lack long-horizon dependencies. Similarly, Wang et al. (2024) used the SoftGym environment (Lin et al., 2020), which includes a suite of short-horizon robotic arm manipulation tasks, as well as MetaWorld environment (Yu et al., 2019), which lacks strict ordering constraints on subgoals, eventually leading to a lack of long-horizon dependencies.

Importantly, VLM reward models are learned-based reward models and, as such, inherently suffer from noisy reward signals. Sadly, the tendency to focus on simpler environments, or to sidestep the challenges of noisy reward signals, keeps appear in recent work. For example, Chan et al. (2023) explored CLIP-based rewards to reduce dependence on human-engineered reward functions, but their experiments were confined to simple housekeeping tasks with small action and state spaces — far less complex than pixel-based environments like Montezuma's Revenge. Similarly, ELLM (Du et al., 2023) relied on a **hard-coded oracle reward model** (labeled as a "VLM reward model" for conceptual demonstration) to maintain training stability. Notably, without this oracle, their agent performed worse than a pure RL agent without any reward shaping — highlighting the limitations imposed by noisy VLM-based rewards in complex environments.

Despite the growing interest of using VLM-based reward models in embodied RL, the literature has largely overlooked the detrimental effects of noisy reward signals on learning efficiency. Key questions remain underexplored: how reliable are these models in complex, long-horizon environments, how do false positive and false negative reward noise affect learning speed, and how to maintain effective learning under noisy reward conditions. These gaps highlight a critical area for progress. In this work, we aim to address this gap by examining these issues in depth and proposing solutions to enhance the applicability of language-guided RL in real-world scenarios.

C The Complete Procedure of the VLM-based Reward Model

Language instructions in real-world tasks are rarely singular; they typically form a sequence that guides an agent step-by-step. A VLM-based reward model must therefore include a mechanism to transition between instructions as the agent progresses. Existing VLM-reward implementations applied a pointer mechanism to decide which instruction the agent should consider at the current step. We follow this implementation and denote the pointer as $m(t)$. It indicates the instruction that the agent is trying to complete at time step t. $m(t)$ is updated according to the following rule:

$$m(t+1) = \begin{cases} 1 & \text{if } t = 0 \\ m(t)+1 & \text{if instr. } l_{m(t)} \text{ completed at } t \\ m(t) & \text{otherwise} \end{cases} \tag{7}$$

The pointer remains on the current instruction until the accumulated reward for completing $l_{m(t)}$ reaches a predetermined threshold.

Given a sequence of instruction sentences $L = \{l_1, l_2, \ldots, l_n\}$, a typical VLM-based reward model, as seen in Pixl2R (Goyal, Niekum, and Mooney, 2020) and ELLM (Du et al., 2023), maintains this **pointer** to the current instruction, starting with l_1. To track progress, it imposes a maximum reward cap — for example, 2.0 — on each instruction. Once the cumulative reward reaches this cap, the pointer advances to the next instruction in the sequence. This approach ensures the agent is incentivized to complete one subtask before moving on to the next.

The complete procedure for training an RL agent with a VLM-based reward model is detailed in algorithm 1. This algorithm builds on concepts including MDP formulation, RL algorithms, and the VLM-based reward model (Appendix B).

Algorithm 1: Instruction-following RL training with VLM-based reward model

Initialize policy network π_θ
Initialize value network Q_ϕ
Setup VLM-based reward model $E(\cdot)$
Split instruction essay into sentences $\{l_1, l_2, ..., l_K\}$
Initialize instruction pointer $p = 1$
Initialize cumulative VLM reward $r_{\text{cum}} = 0$
Initialize cumulative VLM reward threshold q
Initialize replay buffer $\mathcal{D}$
Initialize agent trajectory memory queue τ with length W
for each episode **do**
    Initialize state s_0
    for $t = 0$ to $T - 1$ **do**
        Select action $a_t \sim \pi_\theta(a_t|s_t)$
        Execute a_t, observe next state s_{t+1} and environmental reward r_t^e
        Enqueue (s_t, a_t, r_t, s_{t+1}) in τ
        ▷ *Compute VLM reward*: ◁
        $r_t^v = p(l_p|\tau) = \frac{E(\tau)\cdot E(l_p)}{\|E(\tau)\|\|E(l_p)\|}$
        Combine rewards: $r_t = r_t^e + (1-\beta)\gamma r_t^v$ ▷ β *is a scaling factor*
        Store (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$
        $r_{\text{cum}} \leftarrow r_{\text{cum}} + r_t^v$
        if $r_{\text{cum}} \geq q$ **then**
            $p \leftarrow \min(p+1, K)$
            $r_{\text{cum}} \leftarrow 0$
        if Reach Update Frequency **then**
            Sample mini-batch $\{(s_j, a_j, r_j, s_{j+1})\}$ from $\mathcal{D}$
            ▷ *Compute TD errors*: ◁
            $\delta_j = r_j + \gamma Q_\phi(s_{j+1}, a_{j+1}) - Q_\phi(s_j, a_j)$
            ▷ *Update value network*: ◁
            $\phi \leftarrow \phi + \alpha_v \sum_j \delta_j \nabla_\phi Q_\phi(s_j, a_j)$
            ▷ *Update policy network*: ◁
            $\theta \leftarrow \theta + \alpha_p \sum_j Q(s_j, a_j) \nabla_\theta \log \pi_\theta(a_j|s_j)$

D Convergence time on a sparse-reward landscape

In this work, Section 4, we have shown that the convergence time of a policy learning algorithm on a sparse-reward landscape can be characterized by the distance in the parameter space. Here, we provide a detailed proof of the convergence time on a sparse-reward landscape. We can pin down a characteristic convergence time on a sparse-reward landscape. The sparse-reward setting enforces that

$$r^e(s_t) = \begin{cases} 1 & \text{if } s_t \in \mathcal{S}_G \\ 0 & \text{otherwise} \end{cases}. \tag{8}$$

It is clear from the definition that a good trajectory must always be a goal trajectory, and the cumulative reward is simply the reward at the final goal state so $G^e(\tau) = \gamma^T$ in this setting. For a randomly initialized policy, it is highly unlikely that the initial distribution of trajectories contains any goal trajectory due to the sparsity of goal states. The optimization of $V^e_{\pi_\theta}$ thus consists of two parts. The first part is to search for a goal trajectory. The gradient landscape is almost 0 everywhere, except for cases where a trajectory is δ-close to a goal trajectory. Here δ is the differential unit in the numerical differentiation used in the gradient calculation

$$\theta = \theta + \alpha \nabla_\theta V_{\pi_\theta} \tag{9}$$

such that δ-close means being numerically accessible within a distance of $|\delta|$ in parameter space. And the second part is to reduce T so that goal trajectories become good trajectories and consequently achieving acceptable policies. For the first part, searching for a trajectory for the target is effectively a random walk in the d-dimensional parameter space due to the flat gradient landscape.

Lemma D.1. *For a random walk in n-dimensional space, the expected number of steps T_D needed to travel a distance of D scales with D^2.*

Proof. Let $\vec{X_1}$, $\vec{X_2}$, ... $\vec{X_T}$ be IID random unit vectors uniformly distributed on a $(d-1)$-dimensional sphere $\mathcal{S}^{d-1} \subset \mathbb{R}^d$, where $\vec{X_i} = (X_{i1}, ..., X_{id})$ and $|\vec{X_i}|^2 = \sum_{j=1}^{d} X_{ij}^2 = 1$. Let $\vec{S_T} := \sum_{i=1}^{T} \vec{X_i}$. By a n-dimensional cosine rule we have

$$|\vec{S_T}|^2 = |\vec{S_{T-1}}|^2 + 2\vec{S_{T-1}} \cdot \vec{X_T} + |\vec{X_T}|^2, \tag{10}$$

and because $\mathbb{E}[\vec{X_T}] = \vec{0}$

$$\begin{aligned}
\mathbb{E}[|\vec{S_T}|^2] &= \mathbb{E}[|\vec{S_{T-1}}|^2] + \mathbb{E}[2\vec{S_{T-1}} \cdot \vec{X_T}] + 1 && (11)\\
&= \mathbb{E}[|\vec{S_{T-1}}|^2)] + 2\vec{S_{T-1}}\mathbb{E}[\vec{X_T}] + 1 && (12)\\
&= \mathbb{E}[|\vec{S_{T-1}}|^2] + 1 && (13)\\
&= \mathbb{E}[|\vec{S_{T-2}}|^2] + 1 + 1 && (14)\\
&\vdots && (15)\\
&= T && (16)
\end{aligned}$$

i.e. $\mathbb{E}[|\vec{S_T}|] \sim \sqrt{T}$. When a policy is randomly initialized with θ_0, the distance D to a goal policy, $D := ||\theta_{goal} - \theta_0||$ is fixed and is the distance the random walk needs to travel ($\mathbb{E}[|\vec{S_T}|] = D \sim \sqrt{T_D}$), so the characteristic time needed to travel this distance $T_D \sim D^2$ as we have shown above. □

Assumption D.2. Expert knowledge can guide the parameter search along a path in parameter space, defined by a sequence of n intermediate parameter vectors $\theta_1, \ldots, \theta_n$, where each θ_i represents the parameters after learning sub-task l_i. As a result, the overall distance D can be decomposed into segments: $D \approx \sum_{k=1}^{n-1} d_i$, where $d_i = \|\theta_{i+1} - \theta_i\|$.

Auxiliary rewards essentially open the path for a divide-and-conquer approach by introducing intermediate rewards in the learning process. We introduce BiMI rewards as

$$r^v_{\text{BiMI}}(\tau, l_k) = \max(\mathbf{1}_{\{p(l_k|\tau) \geq \hat{q}\}} - p(l_k), 0) \tag{17}$$

and the cumulative reward of a single trajectory in the presence of BiMI rewards becomes

$$G^v_{\text{BiMI}}(\tau) = \sum_{t=0}^{T} \gamma^t r^v_{\text{BiMI}}(\tau, l_{m(t)}). \tag{18}$$

Because r^v_{BiMI} is either $1-p(l_k)$ or 0, it effectively breaks the entire task into n segments of sub-tasks $\{l_1, l_2, ..., l_n\}$ and each sub-task is a sparse-reward problem. Because this decomposition is based on expert knowledge, we can reasonably assume that the start-finish distance D in parameter space is partitioned into $D \approx d_1 + d_2 + ... + d_{n-1}$ without incurring much detour.

Proposition D.3. *The sum of expected time for a series of random walks, each covering the shorter distance of an individual sub-task, is less than the expected time to travel the entire distance D in one long random walk:* $\frac{1}{n-1}\mathbb{E}[T_D] \leq \mathbb{E}\left[\sum_{i=1}^{n-1} T_{d_i}\right] < \mathbb{E}[T_D]$.

Proof. The expected time taken for each of the sub-tasks then scales with d_i^2 respectively (Lemma D.1), and we have

$$\begin{aligned}
d_1^2 + d_2^2 + ... + d_{n-1}^2 &< (d_1 + d_2 + ... + d_{n-1})^2 && (19)\\
\mathbb{E}\left[\sum_{i=1}^{n-1} T_{d_i}\right] &< \mathbb{E}[T_D] && (20)
\end{aligned}$$

because $d_i > 0 \forall i$. We can also work out that the upper bound for this improvement is a factor of $n-1$ by invoking the Cauchy–Schwarz inequality $\left(\sum_{i=1}^{n} u_i^2\right)\left(\sum_{i=1}^{n} v_i^2\right) \geq \left(\sum_{i=1}^{n} u_i v_i\right)^2$:

$$\begin{aligned}
(d_1^2 + d_2^2 + ... + d_{n-1}^2)(1^2 + 1^2 + ... + 1^2) &\geq (d_1 + d_2 + ... + d_{n-1})^2 && (21)\\
(d_1^2 + d_2^2 + ... + d_{n-1}^2) &\geq \frac{1}{n-1} D^2 && (22)\\
\mathbb{E}\left[\sum_{i=1}^{n-1} T_{d_i}\right] &\geq \frac{1}{n-1}\mathbb{E}[T_D] && (23)
\end{aligned}$$

and the equality sign holds (indicating maximal improvement) when $d_1 = d_2 = ... = d_{n-1}$. The intuitive interpretation is that the divide-and-conquer approach is the most effective when the task is divided evenly into subtasks. □

E VLM-based reward model is a heuristic function

Proposition E.1. *In sparse reward settings, VLM-based reward models, as implemented in Algorithm 1, can be viewed as a heuristic function $h(s_t)$ in HuRL that estimates $V^*(s_t)$.*

Proof. Consider a sparse reward environment where the external reward is defined as:

$$r^e(s_t) = \begin{cases} 1 & \text{if } s_t \in \mathcal{S}_G, \\ 0 & \text{otherwise,} \end{cases}$$

with discount factor $\gamma \in (0, 1)$. For an optimal policy π^*, the value function at state s_t is:

$$V^*(s_t) = \gamma^{\widetilde{T}-t},$$

where $\widetilde{T}$ is the time to reach a goal state $s_g \in \mathcal{S}_G$ along the shortest path from s_0, and $\widetilde{T} - t$ is the remaining steps from s_t. This follows because the agent receives a single reward of 1 at s_g, discounted back to s_t.

In Algorithm 1, the reward $r^v(\tau_t, l_{m(t)}) \in [0, 1]$ measures alignment between the agent's transition (τ_{t-1}, s_t, a_t) and the expert instruction $l_{m(t)}$ for sub-task $m(t)$. By design, $r^v = 1$ when the transition fully matches $l_{m(t)}$, and $r^v < 1$ otherwise, implying *partial alignment*.

We assume expert instructions $\{l_1, \ldots, l_n\}$ define a trajectory τ^* that reaches $\mathcal{S}_G$ optimally or near-optimally. If the agent fully follows $l_{m(t)}$ at each step, its path aligns with τ^*, and $\widetilde{T}$ is minimized (say, $\widetilde{T}^*$). When the agent perfectly follows the instruction at each step along τ^*, the VLM reward is maximized, i.e., $r^v(\tau_t, l_{m(t)}) = 1$. Crucially, moving along the optimal trajectory τ^* leads to the optimal value function $V^*(s_t) = \gamma^{\widetilde{T}^*-t}$.

Now, if the agent deviates from $l_{m(t)}$, it takes a suboptimal action, increasing the expected path length to $\mathcal{S}_G$. Let $\widetilde{T}' > \widetilde{T}^*$ denote this detour. Then, $V^*(s_t) = \gamma^{\widetilde{T}'-t} < \gamma^{\widetilde{T}^*-t}$. Since r^v decreases with misalignment (e.g., from 1 to some $0 \leq r < 1$), it exhibits a **monotonic relationship** with the value function: higher alignment (larger r^v) corresponds to movement towards a shorter path and thus a larger $V^*(s_t)$, and vice versa.

Thus, VLM-based rewards, under Algorithm 1, function as a heuristic $h(s_t)$ that approximates the relative ordering of states in terms of their optimal value $V^*(s)$. □

F False Positive and the Violation of Pessimistic Property of Heuristics

Proposition F.1. *Even if the heuristic remains conservative for all successor states, a single **false positive** overestimation, i.e., $h(s) > V^*(s)$, can violate the pessimistic condition by causing $max_a(\mathcal{B}h)(s, a) < h(s)$.*

Proof. We want to show that given:

1. for all successor states s': $h(s') \leq V^*(s')$ for an arbitrary s.
2. false positive at current arbitrary state s: $V^*(s) < h(s)$.

The objective is to show that under the above conditions, we obtain $\max_a(\mathcal{B}h)(s, a) < h(s)$.

1. Express the Bellman equation for h: $(\mathcal{B}h)(s, a) = R(s, a) + \gamma \sum_{s'} P(s'|s, a)h(s')$
2. Express the Bellman version of V^*: $V^*(s) = \max_{a \in \mathcal{A}} [R(s, a) + \gamma \sum_{s'} P(s'|s, a)V^*(s')]$.
   Given that $h(s') \leq V^*(s')$ for all s', we can expand it to $\sum_{s'} \mathcal{P}(s'|s, a)h(s') \leq \sum_{s'} \mathcal{P}(s'|s, a)V^*(s')$. Therefore, we have:

$$R(s, a) + \gamma \sum_{s'} \mathcal{P}(s'|s, a)h(s') \leq R(s, a) + \gamma \sum_{s'} \mathcal{P}(s'|s, a)V^*(s') = Q^*(s, a) \tag{24}$$

3. Taking the maximum over actions to both sides, we have:

$$\max_a \left(R(s, a) + \gamma \sum_{s'} \mathcal{P}(s'|s, a)h(s') \right) \leq max_a Q^*(s, a) \tag{25}$$

$$\max_a (\mathcal{B}h)(s, a) \leq V^*(s) \tag{26}$$

4. Apply false positive condition: Given $V^*(s) < h(s)$, substituting into the above inequality, we have $\max_a(\mathcal{B}h)(s, a) < h(s)$

Implication: Maintaining a pessimistic heuristic is inherently fragile because the introduction of a false positive in any state disrupts the pessimistic condition. □

G False Positive gives unbounded bias in the *optimality gap*

In this work, Section 4, we analyze the impact of false positive rewards on the convergence speed of policy learning based on the optimality gap analysis by Cheng et al. (2021). Here, we provide a detailed proof of the impact of false positive rewards on the convergence speed of policy learning.

We have defined the formal definition of false positive and false negative from both instruction-following (IF) and heuristic perspectives in Section 4.

Theorem 4.10. *Adapting the optimality gap analysis from Cheng et al. (2021), the gap $V^*(s_0) - V^\pi(s_0)$ in heuristic-guided RL decomposes into regret and bias terms. False negatives preserve the upper bound on bias, as they maintain the heuristic's pessimism, i.e., $h(s) \leq V^*(s)$. In contrast, false positives breaks the pessimistic property, strictly increase the bias without an upper bound, thereby leading to a larger optimality gap and slower convergence.*

We analyze the convergence speed of policy learning by examining the optimality gap, which is defined as the difference between the optimal value of the initial state s_0, $V^*(s_0)$, and the value of the initial state under an arbitrary policy π, $V^\pi(s_0)$. Specifically, we focus on deriving an upper bound for this optimality gap. The key intuition is that a smaller upper bound implies faster convergence to the optimal policy, as fewer iterations of policy updates will be required to reach the optimum.

We begin by stating the theorem made by HuRL authors:

Theorem G.1 (Optimality Gap Decomposition (Cheng et al., 2021))**.** *For any policy π, heuristic $h : \mathcal{S} \to \mathbb{R}$, and mixing coefficient $\beta \in [0, 1]$,*

$$V^*(s_0) - V^\pi(s_0) = \text{Regret}(h, \beta, \pi) + \text{Bias}(h, \beta, \pi) \tag{27}$$

where the regret and the bias term are expressed as follows:

$$\text{Regret}(h, \beta, \pi) \coloneqq \beta \left(\widetilde{V}^*(s_0) - \widetilde{V}^\pi(s_0) \right) + \frac{1-\beta}{1-\gamma} \left(\widetilde{V}^*(d^\pi) - \widetilde{V}^\pi(d^\pi) \right) \tag{28}$$

$$\text{Bias}(h, \beta, \pi) \coloneqq \left(V^*(s_0) - \widetilde{V}^*(s_0) \right) + \frac{\gamma(1-\beta)}{1-\gamma} \mathbb{E}_{s,a\sim d^\pi} \mathbb{E}_{s'\sim \mathcal{P}(\cdot|s,a)} \left[h(s') - \widetilde{V}^*(s') \right] \tag{29}$$

We will not provide a proof for Theorem G.1 in this paper. Please refer to (Cheng et al., 2021) for details. The theorem demonstrates that the performance gap can be elegantly decomposed into two components: a regret term and a bias term such that:

1. The regret term quantifies the difference between $\widetilde{V^*}$ and $\widetilde{V^\pi}$, representing the error caused by π being suboptimal in the reshaped MDP $\widetilde{\mathcal{M}}$. Since π is trained by our selected RL algorithm directly on the reshaped MDP $\widetilde{\mathcal{M}}$, the primary responsibility for minimizing this regret term falls to the RL algorithm itself, not to the design of the auxiliary reward signal. Thus, when evaluating the effects of false positive or false negative rewards, we choose not to focus on bounding the regret term.
2. The bias term captures two key discrepancies: first, between the true optimal value function V^* of the original MDP $\mathcal{M}$ and the optimal value function $\widetilde{V}^*$ of the reshaped MDP $\widetilde{\mathcal{M}}$; second, between $\widetilde{V}^*$ and the heuristic $h(s)$. This term, therefore, reflects the error introduced by addressing the reshaped MDP instead of the original one, alongside how well the heuristic h approximates the optimal value function in the reshaped MDP. Consequently, this bias term directly relates to the quality of the heuristic reward signal h. Hence, we focus on analyzing its upper bound to assess the impact of false positive or false negative rewards on this heuristic.

Cheng et al. (2021) have further derived the upper bound for the bias term, which we present here as a lemma. We omit the proof for brevity; for detailed proof, please refer to (Cheng et al., 2021).

Lemma G.2 (Upper Bound of the Bias Term)**.**

$$\text{Bias}(h, \beta, \pi) \leq (1-\beta)\gamma \left(\mathbb{E}_{\rho^{\pi^*}} \left[\sum_{t=1}^{\infty} (\beta\gamma)^{t-1} (V^*(s_t) - h(s_t)) \right] + \mathbb{E}_{\rho^\pi} \left[\sum_{t=1}^{\infty} \gamma^{t-1} (h(s_t) - \widetilde{V}^*(s_t)) \right] \right) \tag{30}$$

where ρ^π denotes the trajectory **distribution** of $s_0, a_0, s_1, ...$ induced by running π starting from s_0.

This upper bound elegantly illustrates a trade-off: if the heuristic h is set too high (overestimation error), it reduces the first term but increases the second term. Conversely, if h is set too low (underestimation error), it increases the first term but reduces the second term. Just by inspection, it is not immediately clear whether overestimation or underestimation results in a better upper bound, as both impact the bias term in opposing ways.

We now prove that underestimation of h (pessimistic h) is better than overestimation (i.e., $\exists s_t \in \mathcal{S}, h(s_t) > V^*(s_t)$) for minimizing bias.

Proof. 1. **Define Terms**:

- Let $B_1 = \mathbb{E}_{\rho^{\pi^*}} \left[\sum_{t=1}^{\infty} (\beta\gamma)^{t-1} (V^*(s_t) - h(s_t)) \right]$
- Let $B_2 = \mathbb{E}_{\rho^\pi} \left[\sum_{t=1}^{\infty} \gamma^{t-1} (h(s_t) - \widetilde{V}^*(s_t)) \right]$

2. **Underestimation Case (Pessimistic h)**:

Lemma G.3. *If h is pessimistic with respect to $\mathcal{M}$, $\forall \beta \in [0,1], s \in \mathcal{S}, \widetilde{V}^*(s) \geq h(s)$*

Proof. To begin with, we need to use another lemma from (Cheng et al., 2021), shown as follows:

Lemma G.4 (Bellman equation of reshaped MDP (Cheng et al., 2021))**.** *For any policy π, we have*

$$\widetilde{V}^{\pi}(s_0) - h(s_0) = \frac{1}{1-\beta\gamma}\mathbb{E}_{\widetilde{d}^{\pi}_{s_0}}[(\widetilde{\mathcal{B}}h)(s,a) - h(s)] \tag{31}$$

*where $\widetilde{d}^{\pi}_{s_0}$ refers to the **discounted average state distribution** of policy π in reshaped MDP $\widetilde{\mathcal{M}}$. It can be expressed as $\widetilde{d}^{\pi}_{s_0} = (1-\widetilde{\gamma})\sum_{t=0}^{\infty}\widetilde{\gamma}^t d^{\pi}_t$, and d^{π}_t is the state distribution of policy π at time t with $d^{\pi}_0 = \mathbf{1}\{s = s_0\}$.*

For brevity, we do not include the proof for Lemma G.4.
First of all, due to the definition of optimal value V^*, we have

$$\widetilde{V}^*(s_0) \geq \widetilde{V}^{\pi}(s_0) \tag{32}$$

Then, according to Lemma G.4, we can get

$$\widetilde{V}^*(s_0) \geq \widetilde{V}^{\pi}(s_0) = h(s_0) + \frac{1}{1-\beta\gamma}\mathbb{E}_{\widetilde{d}^{\pi}_{s_0}}[(\widetilde{\mathcal{B}}h)(s,a) - h(s)] \tag{33}$$

Recall that the use of VLM-based reward model transform the original MDP into a shaped MDP $\widetilde{\mathcal{M}} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, s_0, \tilde{r}, \gamma \rangle$, where the reward function becomes $\tilde{r} = r^e(s,a) + r^v(\tau_t, l_{m(t)})$. Also, it is important to distinguish the Bellman backup equation under the two different MDPs: the original $\mathcal{M}$ and the reshaped $\widetilde{\mathcal{M}}$. By definition, $(\mathcal{B}h)(s,a) = r(s,a) + \gamma\mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)}[h(s')]$. In contrast, $(\widetilde{\mathcal{B}}h)(s,a) = \widetilde{r}(s,a) + \widetilde{\gamma}\mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)}[h(s')]$. Nevertheless, the two Bellman backup equations possess a remarkable property – they are essentially equivalent to each other:

$$\begin{aligned}
(\widetilde{\mathcal{B}}h)(s,a) &= \widetilde{r}(s,a) + \widetilde{\gamma}\mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)}[h(s')] \\
&= \big(r(s,a) + (1-\beta)\gamma\mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)}[h(s')]\big) + \beta\gamma\mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)}[h(s')] \\
&= r(s,a) + \gamma\mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)}[h(s')] \\
&= (\mathcal{B}h)(s,a)
\end{aligned} \tag{34}$$

Let π denote the greedy policy of $\arg\max_a(\mathcal{B}h)(s,a)$ and then trace back to Equation 34, we have $(\widetilde{\mathcal{B}}h)(s,a) = (\mathcal{B}h)(s,a)$, that means

$$\widetilde{V}^*(s_0) \geq \widetilde{V}^{\pi}(s_0) = h(s_0) + \frac{1}{1-\beta\gamma}\mathbb{E}_{\widetilde{d}^{\pi}_{s_0}}[(\widetilde{\mathcal{B}}h)(s,a) - h(s)] \tag{35}$$

$$= h(s_0) + \frac{1}{1-\beta\gamma}\mathbb{E}_{\widetilde{d}^{\pi}_{s_0}}[(\mathcal{B}h)(s,a) - h(s)] \tag{36}$$

$$\geq h(s_0) \quad \text{[direct result of } h \text{ being pessimistic]} \tag{37}$$

Generalization to Any State: The lemma from (Cheng et al., 2021) (Lemma G.4) is stated in terms of any starting state s_0. Therefore, we can replace s_0 with any state $s \in \mathcal{S}$ in our analysis. Thus we get $\forall s \in \mathcal{S}, \widetilde{V}^*(s) \geq h(s)$ □

Lemma G.3 implies that when h is pessimistic, $B_2 = \mathbb{E}_{\rho^{\pi}}\left[\sum_{t=1}^{\infty}\gamma^{t-1}(h(s_t) - \widetilde{V}^*(s_t))\right] \leq 0$ because $\widetilde{V}^*(s) \geq h(s)$ for all s. Therefore, the bias error is only bounded by B_1 when h is pessimistic, i.e., $\mathrm{Bias}(h,\beta,\pi) \leq B_1$.

3. **Overestimation Case ($\exists s_t \in \mathcal{S}, h(s_t) > V^*(s_t)$)**:
We show briefly that, unlike the underestimation error, the overestimation error does not have a closed-form upper bound expression. The difficulty, as pointed out by Cheng et al. (2021), originates from the trajectory-dependence on $\mathbb{E}_{\rho^{\pi}}[\cdot]$ as the l_{∞} approximation error here can be difficult to control in large state spaces. It is possible that, upon picking up falsely high h states, ρ^{π} gets further distorted away from ρ^{π^*} and accumulates more falsely high h states. In other words, this trajectory dependence makes B_2 prone to a feedback loop of accumulating overestimation errors and resulting in a much larger upper bound for the bias.

□

G.1 Convergence Guarantee of HuRL Compared With Potential-Based Reward Shaping

Previous work on potential-based reward shaping (PBRS) has established that learning a policy on a reshaped MDP can converge to the optimum of the original MDP, as proved by Ng, Harada, and Russell (1999). They proved that the optimal Q value of the reshaped MDP is equivalent to the optimal Q value of the original MDP minus a state-dependent function. Consequently, for an optimal policy defined as $\pi^* = \arg\max_{a \in \mathcal{A}} Q^*(s, a)$, this additional state-dependent value does not alter the policy. Thus, the optimal policy of the original MDP $\mathcal{M}$ is also the optimal policy for the reshaped $\widetilde{\mathcal{M}}$. However, the theoretical convergence for the reshaped MDP within the framework of Heuristic-Guided Reinforcement Learning (HuRL) has not been proven with the same rigor.

The authors of HuRL made a trick which involves manipulating the coefficient β, which scales the heuristic-based auxiliary rewards. If β increases gradually from 0 to 1 over the training process, the agent effectively transitions from interacting with the modified MDP back to the original MDP. This method ensures that, at the end of training, the agent is exactly solving the original problem. Moreover, according to the Blackwell optimal property (Chen, Kochenderfer, and Spaan, 2018), convergence can occur before β reaches 1, thereby allowing the trained policy to maintain optimality in the original MDP under HuRL.

However, in the original HuRL paper, among the five test environments, the β hyperparameter was actually fixed for four of them. Surprisingly, HuRL still converged to the optimal policy faster than the original MDP without updating β. This observation poses an enigma within the HuRL framework, as it is unclear why this convergence to the optimum of the original MDP occurs without dynamically adjusting β.

Thus, while the practical success of HuRL in these environments is evident, the underlying reasons for such convergence remain to be fully understood. This discrepancy between theoretical expectation and empirical results leaves room for future research to explore why HuRL converges under fixed β. Nevertheless, proving the convergence properties goes beyond the scope of our paper, which aims to highlight the prevalence of false positive rewards and their impact. An alternative perspective on this issue is to consider the learning process of maximizing auxiliary rewards as akin to performing behavior cloning from experts. In this sense, we do not need to focus on convergence within the current MDP but can view it as a form of pretraining.

H Additional Experimental Setup Details

H.1 VLM-based Reward Model Procedure

We provide a brief overview of the VLM-based reward model procedure. For a detailed procedure, please refer to Appendix C. The VLM-based reward model will have a pointer to the sequence of the instruction sentence, starting at the first sentence. For original models *Pixl2R* and *ELLM-*, we follow the setting in their original work where for each instruction sentence (the full instruction essay will be split into multiple sentences and treat each sentence as atomic instruction l_k), the reward model will have a maximum cap of rewards (2.0) it can assign to the agent in one episode. When the cap is reached, the reward model will move its pointer to the next instruction sentence. This cap value was selected as a hyperparameter in prior work, intended to serve as a heuristic threshold indicating instruction completion. However, we note that this approach does not guarantee semantic completion, as high cumulative similarity scores can still occur despite incomplete or incorrect execution. For the BIMI reward model, the reward model will move its pointer to the next instruction sentence when the binary signal is triggered.

In terms of the compute resources, we mainly use one NVIDIA 3090 GPU (24GB VRAM) for training and running both the VLM-based reward model and the RL policy model.

H.2 Finetuning VLM-based Reward Models

In contrast to previous work where they rely on hand-crafted oracle multimodal reward models (e.g., Du et al. (2023)), we use actual pretrained VLMs to generate reward signals. 2 VLM backbone models are used in our experiments: 1) *CLIP* (Radford et al., 2021), pretrained by image-text pairs; and (2) *X-CLIP* (Ma et al., 2022), pretrained by video-text pairs. In particular, *Pixl2R* uses *CLIP* because it only uses the single latest frame as input. In contrast, *ELLM-* takes a slice of trajectory (i.e., multiple frames) as input, and thus uses either *X-CLIP* or *CLIP* with additional RNN encoder as the reward model.

Due to the cartoonish and abstract visuals of the testing environments, we further fine-tune the VLMs to adapt to this new visual domain. This fine-tuning is based on the *contrastive learning* method. To collect data, we use well-trained expert agents from (Moon et al., 2023) to generate expert trajectories for the Crafter environments and annotate them with instructions using internal information from the game engine. For Minigrid environments, we use classical search-based planning robots to generate expert trajectories and annotate them with the corresponding task instructions. For Montezuma's Revenge, we manually annotate the expert trajectories.

For Minigrid and Crafter, we collected approximately 80,000 training pairs, each consisting of a natural language instruction and a corresponding visual observation (i.e. a short trajectory snippet). These annotations were generated using internal game state information with corresponding instruction templates. For Montezuma's Revenge, we manually created around 300 high-quality instruction-observation pairs based on expert demonstrations. These training data are of high quality, as we have made every effort to avoid false positive rewards due to poor training data quality. However, despite the fine-tuning process, false positive rewards remain unavoidable. Figure 11 presents an example of instruction data employed for VLM finetuning in the Montezuma's Revenge environment.

```
data_id,instruction,trajectory_chunk_file,trajectory_local_idx
0,climb down the middle ladder,montezuma/expert_traj_chunk_0.pkl,0
1,walk to the right side of the conveyor belt,montezuma/expert_traj_chunk_0
2,jump right to the yellow rope,montezuma/expert_traj_chunk_0.pkl,2
3,jump right to the right platform,montezuma/expert_traj_chunk_0.pkl,3
4,climb down the right ladder,montezuma/expert_traj_chunk_0.pkl,4
5,jump over the skull,montezuma/expert_traj_chunk_0.pkl,5
6,climb up the left ladder,montezuma/expert_traj_chunk_0.pkl,6
7,jump to grab the key,montezuma/expert_traj_chunk_0.pkl,7
8,jump left to the left roof ,montezuma/expert_traj_chunk_0.pkl,8
9,use key to open the left door,montezuma/expert_traj_chunk_0.pkl,9
10,walk left when the laser gate disappears,montezuma/expert_traj_chunk_0.p
11,walk to the middle when the laser gate disappears,montezuma/expert_traj_
12,wait until the laser gate disappears,montezuma/expert_traj_chunk_0.pkl,1
13,approach to the gem,montezuma/expert_traj_chunk_0.pkl,13
14,jump to grab the gem,montezuma/expert_traj_chunk_0.pkl,14
15,walk to the middle when the laser gate disappears,montezuma/expert_traj_
16,climb down the middle ladder,montezuma/expert_traj_chunk_0.pkl,16
17,wait until the spider goes away,montezuma/expert_traj_chunk_0.pkl,17
```

Figure 11: Example of expert instruction data for VLM finetuning under in Montezuma's Revenge environment.

We determine the threshold $\hat{q}$ using conformal prediction to convert continuous cosine similarity scores into binary labels for evaluating the VLM's ability to recognize correct instruction-observation pairs. This allows us to compute standard classification metrics such as precision, recall, and F1-score—providing a preliminary assessment of VLM performance independent of downstream RL interaction. The performance of the fine-tuned VLM-based reward models is shown in Table 4. However, this high precision does not guarantee robust agent performance. As we demonstrate later, agents guided by these models significantly underperform in out-of-distribution (O.O.D.) testing environments, where false positive rewards become prevalent due to distributional shifts, revealing a critical limitation in generalization.

Table 4: Performance of fine-tuned VLM reward model on the testing dataset using the 90th percentile empirical quantile as threshold

Environment	Precision	Accuracy	F1 Score	Recall	Model
Crafter	0.9847	0.9466	0.8538	0.9702	CLIP ELLM-
Crafter	0.9799	0.9028	0.7618	0.9842	CLIP Pixl2R
Crafter	0.2095	0.2514	0.2868	0.9657	XCLIP ELLM-
Minigrid	0.7260	0.9200	0.7849	0.9763	CLIP ELLM-
Minigrid	0.6992	0.9086	0.7592	0.9616	CLIP Pixl2R
Minigrid	0.1716	0.2310	0.2642	0.9704	XCLIP ELLM-
Montezuma	0.8838	0.9638	0.8825	0.9478	CLIP ELLM-
Montezuma	0.8343	0.9108	0.7652	0.9842	CLIP Pixl2R
Montezuma	0.8044	0.9259	0.8045	0.9657	XCLIP ELLM-

H.3 Hyperparameters for VLM Reward Model + RL Agents

In the experiments, all methods are implemented based on PPO with same model architecture. The Minigrid and Crafter environments use the same training hyperparameters as the Achievement Distillation paper (Moon et al., 2023). For Montezuma's Revenge, we found that the performance of the agent was sensitive to the gamma and GAE lambda parameters. To improve the performance of agents in Montezuma's Revenge, we took two additional steps: (1) normalizing the observation inputs when computing the rewards, and (2) not normalizing the advantage during the GAE calculation. The hyperparameters are shown in Table 5, Table 6 and Table 7.

Table 5: Agent Policy Model Parameters

Parameter	Value
model_cls	Recurrent PPO
hidsize	1024
gru_layers	1
impala_kwargs	
- chans	[64, 128, 128]
- outsize	256
- nblock	2
- post_pool_groups	1
- init_norm_kwargs	
- batch_norm	false
- group_norm_groups	1
dense_init_norm_kwargs	
- layer_norm	true

Table 6: Crafter and Minigrid RL Training Parameters

Parameter	Value
gamma	0.95
gae_lambda	0.65
algorithm_cls	PPO Algorithm
algorithm_kwargs	
- ppo_nepoch	3
- ppo_nbatch	8
- clip_param	0.2
- vf_loss_coef	0.5
- ent_coef	0.01
- lr	3.0e-4
- max_grad_norm	0.5
- aux_freq	8
- aux_nepoch	6
- pi_dist_coef	1.0
- vf_dist_coef	1.0

Table 7: Montezuma's Revenge RL Training Parameters

Parameter	Value
gamma	0.99
gae_lambda	0.95
int_rew_type	"rnd"
pre_obs_norm_steps	50
algorithm_cls	PPO Algorithm
algorithm_kwargs	
- update_proportion	0.25
- ppo_nepoch	3
- ppo_batch_size	256
- clip_param	0.1
- vf_loss_coef	0.5
- ent_coef	0.001
- lr	1.0e-4

H.4 Evaluation Metrics

As detailed in Appendix A.1, we adopted the ***score*** metric from the Crafter benchmark (Hafner, 2022) for performance evaluation, as it effectively measures consistent performance across multiple subtasks in sparse reward environments. Unlike the *maximum total rewards* metric, which does not adequately reflect consistent performance, the score metric offers a more reliable indicator of learning progress.

I Details of Showing the Prevalence of False Positives in VLM Cosine Similarity Scores

In this work, we have shown that the VLM-based reward models assign high rewards to manipulated trajectory-instruction pairs, indicating the prevalence of false positive rewards. We list the figures of reward signals from learned VLMs for different types of trajectory-instruction pairs for each individual environment — the Montezuma, Minigrid, and Crafter environments. All figures show that the VLM-based reward models assign high rewards to manipulated trajectory-instruction pairs, indicating the prevalence of false positive rewards.

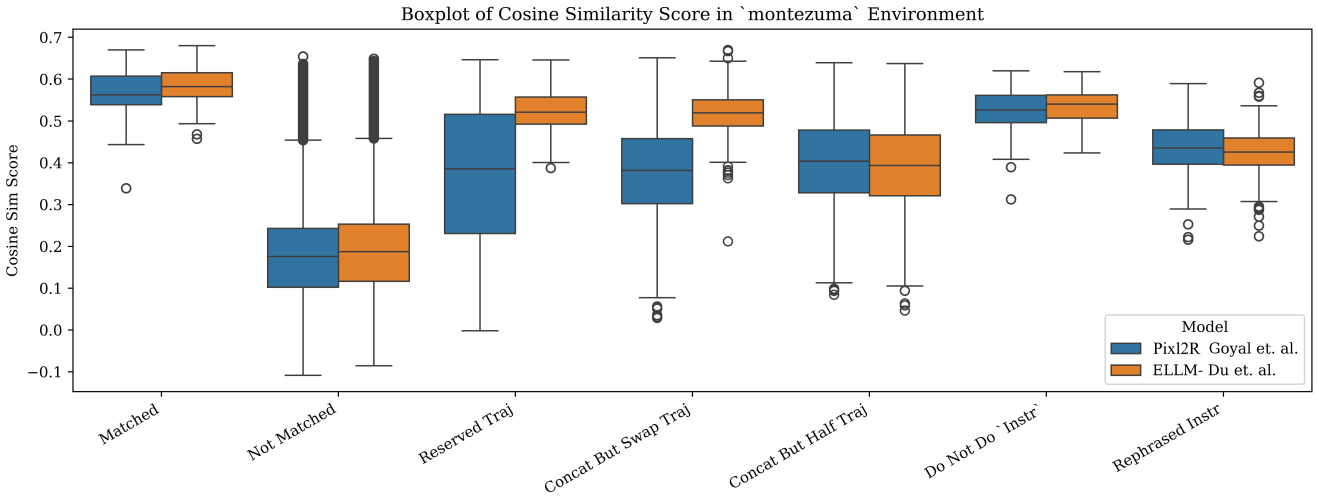


Figure 12: Cosine similarity scores for match, mismatch and manipulated trajectory-instruction pairs in Montezuma.

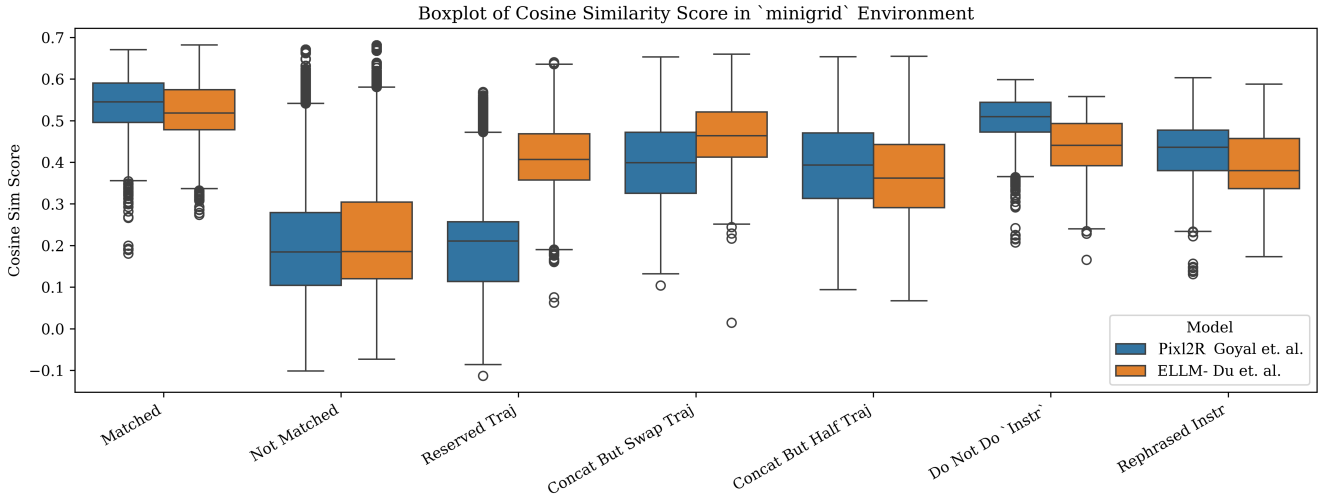


Figure 13: Cosine similarity scores for match, mismatch and manipulated trajectory-instruction pairs in Minigrid.

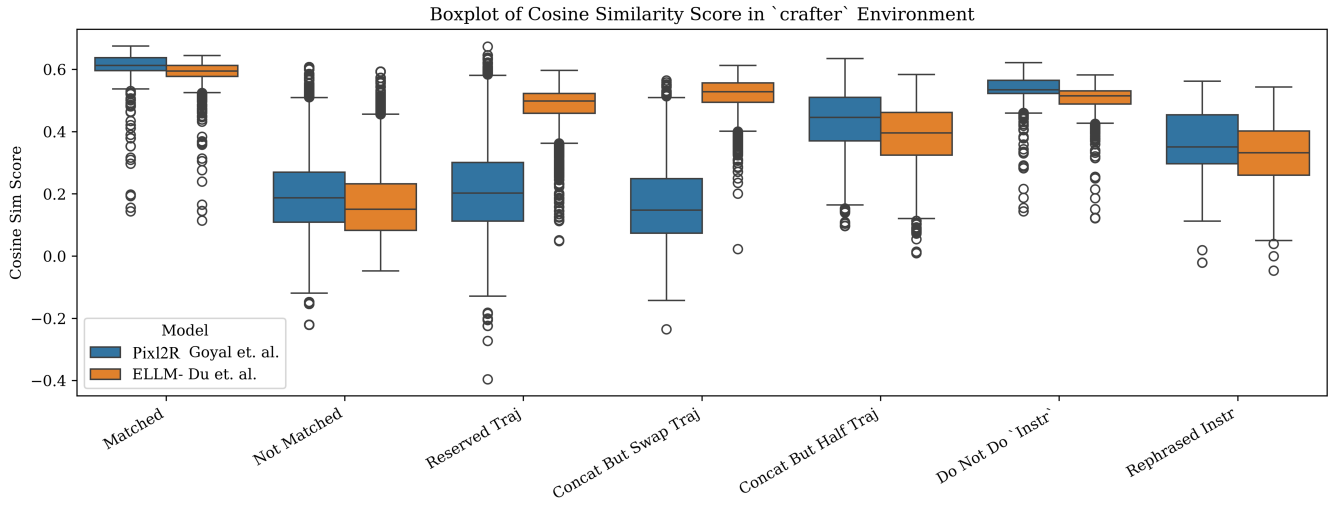


Figure 14: Cosine similarity scores for match, mismatch and manipulated trajectory-instruction pairs in Crafter.

J Extra Experiment on Simulated Oracle Reward Model to demonstrate H3 and H4

In our efforts to assess the impact of false positive rewards from auxiliary reward model without the interference of other factors such as domain shift, poor data quality, and errors from other issues such as the choice of multimodal architectures, we devised a **simulated** auxiliary reward model (also known as *oracle* auxiliary reward model) that access to internal state information from the game engine. The model compares the sequence of past actions and states of the agent with predefined target intermediate state sets that map to each instruction sentence. This is feasible in Montezuma's Revenge environment as we are able to access coordinate system information directly from the game engine. This access allows us to locate the current positional information of the agent and also label specific intermediate states as targets and assign rewards to the agent accordingly.

We therefore designed three types of simulated reward model:

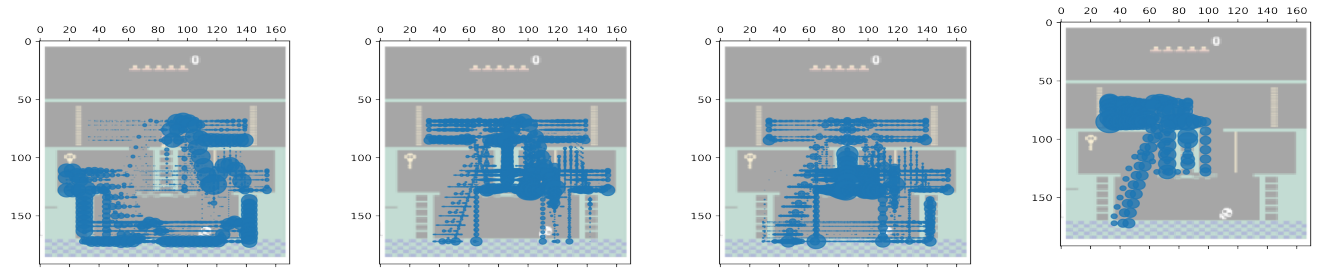
- **Sim RM 1 (Perfect)** generates rewards accurately whenever the agent reaches the designated intermediate states. Furthermore, it strictly adheres to the chronological sequence of instructions; rewards for subsequent instructions are only awarded if all preceding instructions have been fulfilled.
- **Sim RM 2 (False Positive)** introduces a tolerance for false positive rewards but with reduced reward magnitudes, all while maintaining the temporal sequence. This is implemented by defining a radius σ, where if the agent enters a circle centered at the target state with a radius of σ, it receives a small amount of rewards.
- **Sim RM 3 (Temporal Insensitive False Positive)** disregards the chronological order of instructions, allowing rewards for later tasks even if earlier ones remain unfulfilled. However, note that fulfilling every sub-task will still result in the agent receiving the maximum total rewards. Therefore, in theory, the policy will eventually converge.

Results are reported in Table 8. Several important observations are as follows:

Table 8: Agent performance in Montezuma's Revenge, evaluated across three rooms with the goal state being the exit through a designated door. Metrics measured are the Area Under the Curve (AUC) for total reward (where higher is better) and success rate (SR) of reaching the goal state. The baseline is annotated using a $\star$ symbol. The results are averaged over 5 runs, with standard deviations reported.

Model	AUC	SR
(1) PPO (Schulman et al., 2017)	all failed	0%
(2) PPO+RND $\star$ (Burda et al., 2018)	0.550±0.066	100%
(3) PPO + **Sim RM 1 (Perfect)**	0.287±0.048	100%
(4) PPO+RND + **Sim RM 1 (Perfect)**	0.608±0.073	100%
(5) PPO+RND + **Sim RM 2 (False Positive)**	0.183±0.187	73.3%
(6) PPO+RND + **Sim RM 3 (Temp. Insen.)**	0.051±0.116	16.7%

1. Perfect auxiliary reward model have shown enhanced performance compared to weak RL baselines like PPO. While agents trained solely with PPO struggled to play Montezuma, incorporating Perfect auxiliary reward model into PPO (i.e., entry (3)



(a) PPO+RND (same as + RM (False Negative))

(b) PPO+RND + Sim RM 1 (Perfect Simulation)

(c) PPO+RND + Sim RM 2 (False Positive)

(d) PPO+RND + Sim RM 3 (Temporal Insensitivity)

Figure 15: Movement heatmap for PPO+RND agents when different simulated auxiliary instruction-following-based reward models are involved.

in Table 8) did find the goal state, with the success rate increasing from 0% to 100%. However, we observed that the auxiliary reward model learns more slowly than the intrinsic reward model. This is evidenced by its lower AUC score (0.287 for entry (3) vs. 0.550 for entry (2) in Table 8), where AUC reflects how quickly the agent learn to reach the goal state. This finding was initially surprising, but upon examining the agent movement heatmaps in Figure 15a and Figure 15b, an explanation becomes clear: the PPO+RND model discovers shorter paths to the goal state. The heatmap reveals that the PPO+RND agent learns to directly jump to a rope, bypassing the use of a ladder and conveyor belt as suggested by the expert instructions. This observation highlights a nuance in VLM-based reward signals, potentially overlooking more efficient routes when expert instructions do not serve as landmarks (a well-known concept in heuristic search in the field of classical planning).

2. We observed a remarkable synergy between the instruction-based reward model and the intrinsic reward model. As shown in entry (4) in Table 8, the PPO+RND + Sim RM 1 (Perfect) agent achieves the highest AUC score, demonstrating that it learns to reach the goal state faster than any other model.
3. False positive rewards significantly impeded learning, as shown by entries (5) and (6) in Table 8. Heatmaps (Figure 15b, Figure 15c) reveal agents frequently visiting dead ends (e.g., falling off cliffs) under these conditions, suggesting that the agent is trapped in local minima. To assess their impact, we compare these to PPO+RND (entry (2)), a baseline lacking VLM-based instruction rewards. Since (2) does not even have a VLM-based reward model, we treat it as a baseline with **full false negative** instruction-based rewards. The AUC scores of (5) and (6) are lower than (2), indicating that false positives are more detrimental than false negatives, supporting our hypothesis **H4**, as detailed shortly.
4. When the VLM reward model ignored temporal ordering (entry (6)), convergence failed, with the success rate dropping to 16.7% and RND unable to recover within the time limit. Heatmap (Figure 15d) illustrates the agent fixating on the final instruction ("walk left to the door") without securing the key, trapping it in local minima near s_0 due to false positives — a form of composition insensitivity (Section 5). Moreover, given that Montezuma's Revenge environment has a fixed initial state distribution, (i.e., the agent always starts at a fixed s_0), it is not guaranteed that the RL algorithm will eventually reach the global optimum with maximum total rewards, as highlighted by Agarwal et al. (2019).

Figure 15 provides initial evidence of the differential impact of false negatives versus false positives on training outcomes, as posited by **H4**, suggesting that false positives — particularly those tied to temporal insensitivity — more severely degrade final scores.

K Algorithm for Empirical Quantile Calculation

The empirical quantile $\hat{q}$ is calculated using the conformal prediction method, shown as follows:

Algorithm 2: Calculate Empirical Quantile ($\hat{q}$) in VLM RM

Require: Calibration set $\{\tau, l\}_n$, where l is the instruction sentence, τ is the corresponding trajectory, and n is the number of samples;
User-defined error rate α;
VLM model reward model v

▷ *Obtain the similarity-based score* ◁

$\{r\}_n \leftarrow \{v(\tau, l)\}_n$

▷ *Compute the quantile level* ◁

$q_{\text{level}} \leftarrow \frac{\lceil (n-1)\times(1-\alpha) \rceil}{n}$

▷ *Compute the empirical quantile* ◁

$\hat{q} \leftarrow$ `np.quantile(`$\{r\}_n$`,` $q_{\texttt{level}}$`, method='lower')`

return $\hat{q}$

L Ablation Study Lineplots for BiMI reward

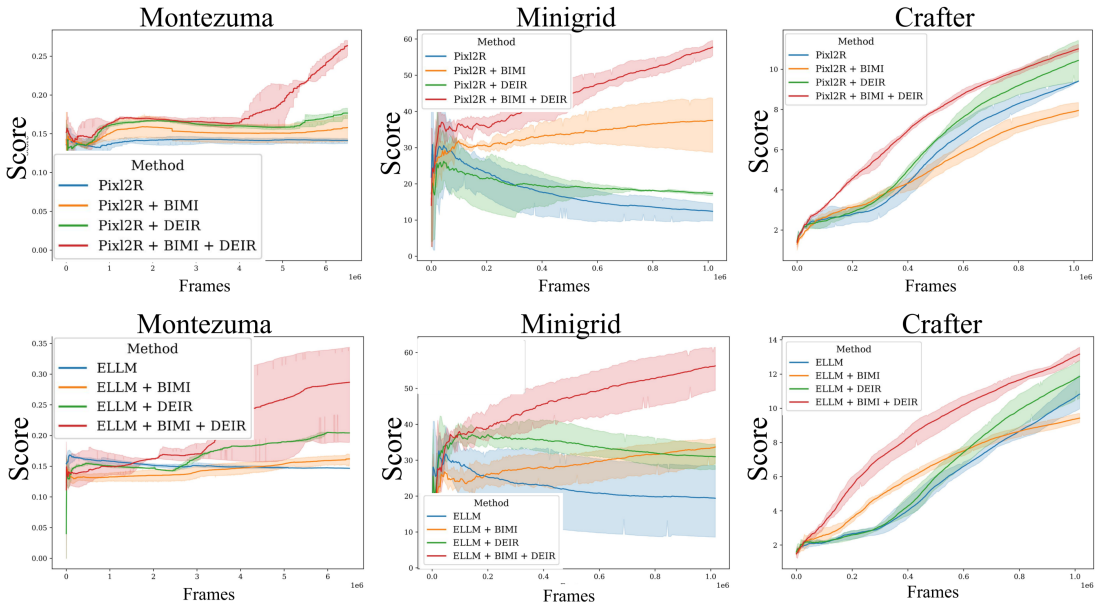


Figure 16: Besides the improvements of the score performance of agents across different environments with the BIMI reward function, it also collaborates well with intrinsic rewards. Combining both can lead to significant performance improvements

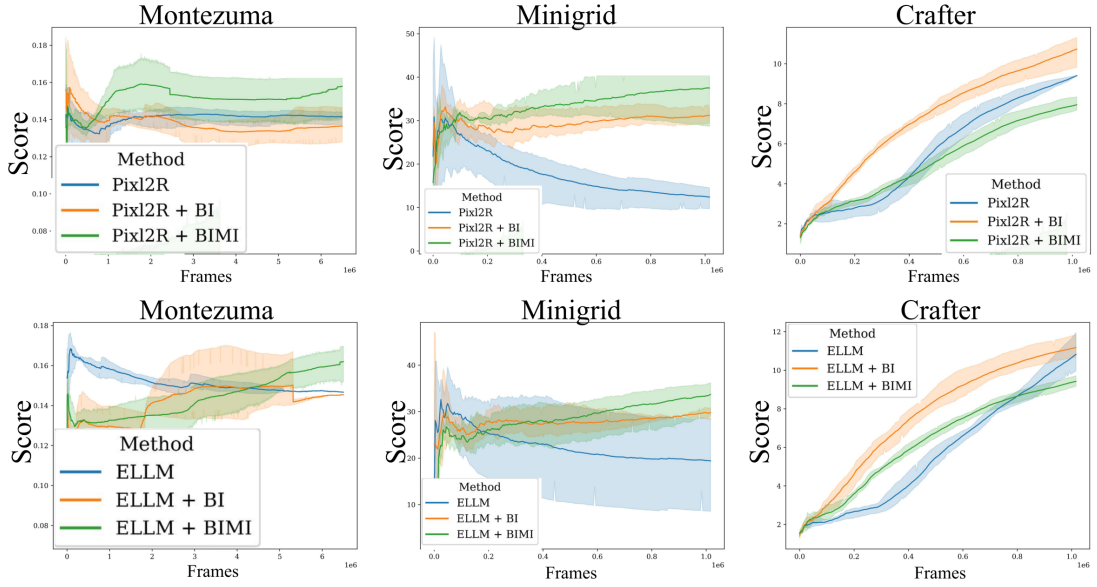


Figure 17: Ablation on the components of BIMI reward function. The binary reward (Bi) alone led to a 36.5% improvement compared to original models. Excluding Crafter, Mutual Information (MI) provided a 23% further improvement over Bi alone

M Encoder-based Vision-Language Models Explanation

Similar to LLMs, VLMs can be broadly classified into two categories: *encoder-based models* (or *dual-encoder models*) and *generative-based models* (often referred to as *Vision LLMs*).

Encoder-based models focus on mapping vision and language data into a shared semantic space where embeddings of corresponding visual and textual content are positioned close to each other. CLIP (Radford et al., 2021) and SigLIP (Tschannen et al., 2025) exemplify this approach. They are often referred to as VLM dual-encoders as such VLMs consists of both vision

encoder and text encoder which can be operated separately to encode vision or language data. Dual-encoder models are particularly well-suited for cross-modal retrieval tasks, such as finding the most relevant image for a given text query or vice versa. This is achieved by computing similarity scores (e.g., cosine similarity) or other distance metrics between the embeddings of the query and the images in the dataset. Dual-encoders are therefore being explored to serve as learned-based reward models for embodied RL agents (Rocamonde et al., 2024; Du et al., 2023), which we discuss in Appendix B.

In contrast, *generative-based models* such as LLaVA (Liu et al., 2023) and Sa2VA (Yuan et al., 2025) are typically built by connecting a vision encoder (e.g., CLIP-based vision encoder) to an LLM via a projection layer (e.g., a Multilayer Perceptron (MLP)). The backbone LLM takes both vision feature vectors and text tokens as input and generates textual outputs, enabling these models to perform tasks such as Vision Question Answering (VQA). Although both categories are commonly classified as VLMs, they serve fundamentally different purposes and exhibit distinct behaviors.

N Sparse Reward and Random Walk

Below, we will discuss the characteristics of RL training under sparse rewards

Proposition N.1. *With sparse rewards, the gradient landscape is nearly flat, making gradient-ascent updates indistinguishable from a random walk in parameter space.*

While this may seem obvious, no existing textbook has yet explained this concept clearly. Here, we aim to formalize and justify this proposition, as it will be crucial for analyzing the performance of VLM-based reward models in this work.

Recall the gradient of the policy gradient update in online Actor Critic methods: $\theta \leftarrow \theta + \alpha \mathbb{E}_{\pi_\theta}\left[Q_\phi(s,a)\nabla_\theta \log \pi_\theta(a|s)\right]$. Using chain rule, the gradient of the policy gradient update can be decomposed as:

$$\mathbb{E}_{\pi_\theta}\left[\frac{Q_\phi(s,a)}{\pi_\theta(a|s)}\nabla_\theta \pi_\theta(a|s)\right]$$

In sparse reward settings, the Q-function $Q^{\pi_\theta}(s,a)$, often approximated by a neural network in actor-critic methods, exhibits significant randomness. Initially, the critic's weights are randomly initialized (e.g., via Xavier initialization (Glorot and Bengio, 2010)), producing arbitrary Q-values (Mnih et al., 2015). With most transitions yielding $r = 0$, the TD error $\delta = \gamma Q(s', a') - Q(s,a)$ is typically non-zero due to the stochastic variation between $Q(s,a)$ and $Q(s',a')$ resulting from random initialization. This leads to random updates to the Q-network, as the noisy TD error lacks a consistent direction, further increasing the stochasticity of Q-value estimates.

Such stochasticity propagates to the policy gradient update, where it even amplifies this effect: At initialization, $\pi_\theta(a|s) \approx 1/K$ for a discrete action space with K actions, as random weights yield a near-uniform softmax distribution. The gradient term $\frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)}$ scales $Q_\phi(s,a)$ by the inverse of the policy probability. Therefore, the update step $\theta \leftarrow \theta + \alpha_\theta \frac{Q_\phi(s,a)\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)}$ has a high variance due to noisy Q-value estimates and non-trivial gradient magnitudes from the scaling effect of $\pi_\theta(a|s)$. Consequently, the update exhibits $\text{Var}(\hat{\nabla}_\theta J(\theta)) > 0$ and $\mathbb{E}[\hat{\nabla}_\theta J(\theta)] \approx 0$.

This behavior renders policy updates indistinguishable from a *random walk* in parameter space. The parameters θ drift stochastically, as each step's direction is dominated by noise rather than progress toward a better policy. This random walk behavior is a key reason why training under sparse rewards is challenging.