PROSE: LEARNING PROGRESS FROM CROSS-TRAJECTORY OUTCOMES FOR LONG-HORIZON EMBODIED TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon embodied tasks are difficult to learn from sparse terminal rewards, because an agent may need to act for hundreds of steps before receiving any feedback about its intermediate behavior. Existing approaches provide intermediate guidance by scoring individual observations based on their relevance to the task goal or by relying on predefined subgoals or dense progress supervision. However, goal relevance alone may be weakly predictive of eventual task success, and the effectiveness of subgoal-based guidance depends on whether they remain appropriate under the current environment dynamics and execution capabilities. Instead of requiring explicit intermediate targets, we identify recurring behavior patterns associated with successful outcomes across trajectories and use them to supervise promising behaviors before terminal feedback arrives. We introduce PROSE (**P**rogress **R**ecognition from **O**utcome **S**tatistics across **E**xperience), which abstracts long trajectories into visually coherent periods, aligns similar periods across episodes into recurring situations and transitions, estimates their progress from cross-trajectory outcome statistics, and refines it along individual trajectories by penalizing subsequently lost progress and accounting for how each transition is visually and behaviorally executed. When progress persistently stalls, a targeted exploration controller uses the agent's previous observations to redirect exploration away from unpromising regions. Across five long-horizon MineDojo tasks, integrating PROSE with two world-model agents improves success rate by 9.1 points and reduces steps to first success by 14.4% on average. These results show that PROSE complements different world-model agents by turning sparse outcomes into intermediate supervision for long-horizon embodied tasks.

1 INTRODUCTION

Embodied agents operating in open-world environments must often solve tasks that require hundreds of interactions before receiving meaningful feedback (Hafner et al., 2023; Li et al., 2024; Yuan et al., 2023). When rewards are sparse, intermediate behaviors receive little or no immediate feedback regardless of whether they advance the task, making it difficult to identify which actions contribute to eventual success (Peng et al., 2026; Verma & Metcalf, 2024; Arjona-Medina et al., 2019; Patil et al., 2020; He et al., 2026), as shown in Fig. 1(a). Model-based reinforcement learning (MBRL) is attractive for long-horizon tasks because learned world models can simulate alternative futures and capture the longer-term consequences of different actions (Ha & Schmidhuber, 2018; Zhou et al., 2024b; Wang et al., 2026; Song et al., 2026; Zhang et al., 2026a). In imagination-based world-model agents, these futures are evaluated using predicted rewards learned from environment interaction (Hafner et al., 2019; 2025). When rewards are sparse and terminal, the predicted rewards provide little intermediate supervision for distinguishing whether imagined behavior is making progress toward eventual success.

Existing approaches provide intermediate guidance by scoring individual observations according to their relevance to the task goal (Li et al., 2025; 2026b). However, goal relevance alone may not reflect whether the current behavior is predictive of eventual task success. For example, when the task is to cut a tree, a tree across an impassable cliff is highly goal-relevant, although approaching it may not improve the chance of eventual completion. We therefore seek a progress signal grounded

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

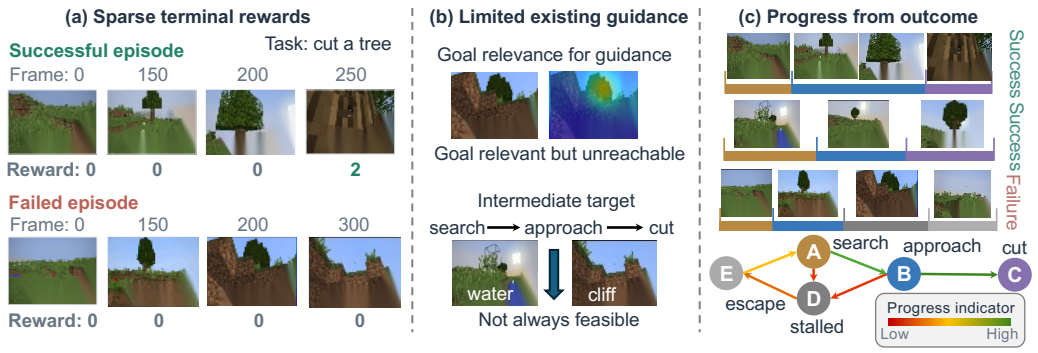


Figure 1: (a) Sparse terminal rewards provide little supervision for intermediate behavior. (b) Goal relevance and explicit intermediate targets may provide limited guidance when feasibility depends on the encountered situation. (c) PROSE aligns recurring situations and transitions across completed trajectories and uses cross-trajectory terminal outcome statistics to estimate intermediate progress.

in trajectory outcomes rather than goal similarity alone. Other approaches use demonstrations or explicit subgoals to provide more intermediate supervision (Escoriza et al., 2025; Kim et al., 2021; Lee et al., 2026; Peng et al., 2026). Such methods may adapt subgoals online, but still formulate intermediate guidance as explicit targets for the agent to reach. In open-world environments, selecting an appropriate target can itself depend on the current environment dynamics and execution capabilities. For example, when a direct route to a tree is blocked, the agent may need to reroute or escape before approaching it (Fig. 1(b)).

In this work, we take a retrospective, cross-trajectory approach to intermediate supervision by identifying recurring behavior patterns associated with eventual outcomes. Based on this idea, we introduce PROSE (**P**rogress **R**ecognition from **O**utcome **S**tatistics across **E**xperience), which abstracts long interaction trajectories into visually coherent periods and aligns similar periods across episodes using semantic and visual-temporal representations to identify recurring situations and transitions. For each recurring transition, PROSE aggregates terminal outcomes across trajectories and measures its association with eventual success relative to the population success rate. This provides a shared cross-trajectory prior for estimating intermediate progress as illustrated in Fig. 1(c).

We further complement the shared cross-trajectory progress estimate with trajectory-specific cues that capture how progress is maintained and executed within individual trajectories. First, an agent may reach transitions strongly associated with success but subsequently lose that progress. We therefore introduce a degradation penalty when the estimated success probability decreases along the graph path, encouraging progress not only to be reached but also maintained. Second, the same recurring transition can be realized through different low-level action sequences, some of which advance the task more effectively than others while receiving the same graph reward. We therefore learn an execution-specific visual-action residual that refines the shared prior using the local visual and action dynamics of the current transition. These refinements produce the final progress reward used to guide imagined behavior learning. If progress persistently stalls during environment interaction, PROSE invokes a targeted exploration controller that tracks previously observed area to redirect the agent away from repeatedly unpromising regions.

We evaluate PROSE on long-horizon MineDojo tasks using two world-model agents, LS-Imagine (Li et al., 2025) and DreamerV3 (Hafner et al., 2023), and observe consistent improvements in task success and completion efficiency. These results suggest that learning intermediate progress from accumulated cross-trajectory outcomes can complement different world-model agents, rather than being tied to a single base method. In summary, our contributions are threefold:

- We introduce PROSE, which infers intermediate progress from recurring situations and transitions across successful and failed trajectories, turning sparse terminal outcomes into progress supervision without predefined subgoals or dense annotations.
- We complement shared cross-trajectory progress estimation with trajectory-specific cues that capture how progress is maintained and executed, through a degradation penalty for

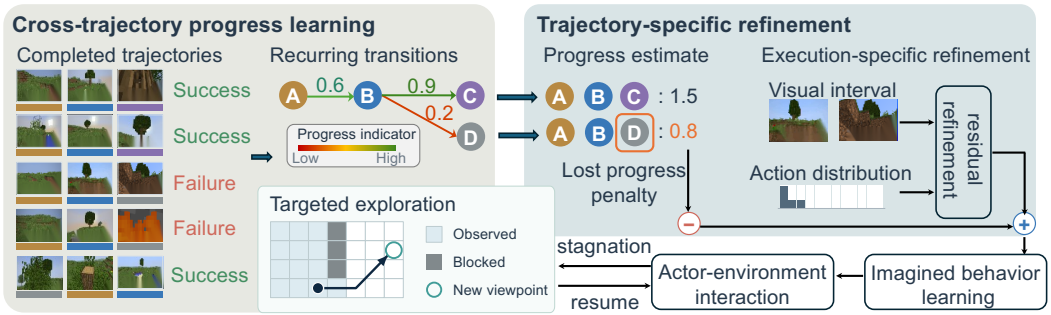


Figure 2: Overview of PROSE. Long trajectories are abstracted into visually coherent periods and aligned across episodes to identify recurring situations and transitions. Cross-trajectory terminal outcome statistics provide a shared prior for intermediate progress, which is refined by a degradation penalty and an execution-specific visual-action residual. The resulting progress reward guides imagined behavior learning, while persistent stagnation triggers targeted exploration using the agent's history of observed, visited, and blocked regions.

lost progress, an execution-specific module for different realizations of same recurring transitions and targeted exploration when progress stalls.

- Across five long-horizon MineDojo tasks, PROSE improves two world-model agents with an average 9.1-point gain in success rate and a 14.4% reduction in steps to first success, showing that cross-trajectory outcome supervision is complementary across agents.

2 METHOD

Given a set of interaction trajectories with sparse terminal outcomes, our goal is to derive intermediate progress rewards that indicate how individual transitions are associated with eventual task completion. PROSE abstracts long trajectories into recurring transitions that can be compared across episodes (Sec. 2.1), uses the terminal outcomes of trajectories containing these transitions to estimate intermediate progress, and refines this estimate to account for progress that is subsequently lost and for differences in how the same transition is executed (Sec. 2.2). Finally, when the resulting progress signal indicates that progress persistently stalls, targeted exploration guides the agent toward alternative observation regions (Sec. 2.3).

2.1 FROM LONG TRAJECTORIES TO RECURRING TRANSITIONS

Long embodied trajectories may contain hundreds of interaction steps and differ substantially in duration. To make cross-trajectory comparison tractable, we first compress each trajectory into visually coherent periods, represent each period using semantic and visual-temporal information, and align recurring periods across trajectories into a shared transition graph.

Temporal abstraction First, we compress each trajectory into a sequence of visually coherent periods. Given a trajectory $\tau = (\{(o_t, a_t)\}_{t=1}^{T}, y)$, where o_t and a_t denote the observation and action at step t, respectively, and $y \in \{0, 1\}$ is the terminal task outcome, we subsample observations at stride five, obtaining $\tilde{o}_1, \ldots, \tilde{o}_N$, with $N \approx T/5$. Each sampled frame is encoded using a frozen MineCLIP visual encoder (Fan et al., 2022), denoted as f_n. We then measure the visual similarity among the sampled frames using cosine similarity,

$$S_{ij} = \frac{f_i^\top f_j}{\|f_i\|_2 \|f_j\|_2}, \qquad i, j \in \{1, \ldots, N\}, \tag{1}$$

yielding a trajectory-level similarity matrix $S \in \mathbb{R}^{N \times N}$. We identify visual change points according to S as the high similarity indicates consistent visual content (details in Appendix A.1). These change points partition the original trajectory into K contiguous periods $\bar{\tau} = (p_1, p_2, \ldots, p_K)$, $K \ll N$, where $p_k = \{(o_t, a_t)\}_{t=\ell_k}^{u_k}$, and ℓ_k and u_k denote the starting and ending time steps of p_k. Each period thus represents a temporally contiguous and visually coherent segment of the trajectory.

Algorithm 1 Progress Graph Node Construction

Input: Construction periods $\mathcal{P}$ with (b_k, h_k), matching threshold δ, minimum trajectory support m.
Output: Retained nodes $\mathcal{V}$ and prototypes $\{\mu_i\}$.

Period assignment and node retention

$\widetilde{\mathcal{V}} \leftarrow \emptyset$
for each $p_k \in \mathcal{P}$ **do**
  $\mathcal{C}_k \leftarrow \{i : z_i \in \widetilde{\mathcal{V}},\ b_i = b_k\}$
  $i \leftarrow \textsc{Nearest}(h_k, \mathcal{C}_k)$
  if $i \neq \perp$ and $\cos(h_k, \mu_i) \geq \delta$ **then**
    $\textsc{UpdateNode}(i, p_k)$
  else
    $\textsc{CreateNode}(p_k)$
  end if
end for
$\mathcal{V} \leftarrow \{z_i \in \widetilde{\mathcal{V}} : |\mathcal{T}_i| \geq m\}$
return $\mathcal{V}, \{\mu_i\}_{z_i \in \mathcal{V}}$

Node updates

procedure $\textsc{UpdateNode}(i, p_k)$
  $v_i \leftarrow v_i + h_k$; $\mu_i \leftarrow \mathrm{Norm}(v_i)$
  $\mathcal{T}_i \leftarrow \mathcal{T}_i \cup \{\mathrm{traj}(p_k)\}$
end procedure

procedure $\textsc{CreateNode}(p_k)$
  Allocate node z_i; $b_i \leftarrow b_k$
  $v_i \leftarrow h_k$; $\mu_i \leftarrow h_k$
  $\mathcal{T}_i \leftarrow \{\mathrm{traj}(p_k)\}$
  $\widetilde{\mathcal{V}} \leftarrow \widetilde{\mathcal{V}} \cup \{z_i\}$
end procedure

Notation: v_i is the descriptor sum, $\mu_i = \mathrm{Norm}(v_i)$ the node prototype, and $\mathcal{T}_i$ the set of supporting trajectory IDs. $\mathrm{traj}(p_k)$ gives the trajectory ID of period p_k. NEAREST returns the cosine-nearest node index, or $\perp$ if no candidate exists.

Period representation For each period, we construct two complementary representations: a coarse semantic descriptor that characterizes the task situation and a visual-temporal descriptor that captures its particular observation. We provide a VLM with up to 20 frames sampled across one period p_k, together with up to 4 frames from p_{k-1} and p_{k+1} for contextual information. The VLM predicts a structured description b_k of the situation. $b_k = b(p_k) = (q, s_k, v_k, c_k, w_k)$, where q denotes the task, and s_k, v_k, c_k, and w_k denote the event stage, target visibility, target accessibility, and view quality, respectively.

To retain finer visual and temporal information that may be lost by this semantic abstraction, we additionally construct a MineCLIP descriptor. We uniformly sample $M = 8$ frames from p_k and extract normalized features $F_k = (f_k^1, \ldots, f_k^M), f_k^m \in \mathbb{R}^{512}$. The ordered features capture the visual sequence within the period, while its start-to-end visual change is represented as

$$d_k = \frac{f_k^M - f_k^1}{\max(\|f_k^M - f_k^1\|_2, \epsilon)}. \tag{2}$$

We set $d_k = 0$ when the difference norm is below ϵ. We combine them with F_k:

$$h_k = \mathrm{Norm}\left(\left[\sqrt{0.8/M}\ \mathrm{vec}(F_k); \sqrt{0.2}\, d_k\right]\right) \in \mathbb{R}^{4608}. \tag{3}$$

Each period p_k is therefore represented by $\{(b_k, h_k)\}$, where b_k captures its coarse semantic situation and h_k preserves its finer visual-temporal realization. These period representations form the units for cross-trajectory alignment in the following section.

Cross-trajectory alignment Given the period representations $\{(b_k, h_k)\}$, we align recurring transitions across trajectories into a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each node $z_i \in \mathcal{V}$ represents a recurring situation shared across period occurrences, and each directed edge represents an observed transition between two such situations. We distinguish p_k, a particular period occurrence within a trajectory, from z_i, which aggregates matched occurrences across trajectories. For a period p_k, candidate nodes are restricted to those sharing its coarse semantic bucket b_k. Its visual-temporal descriptor h_k is then compared with the corresponding node prototypes using cosine similarity. If the nearest prototype exceeds a similarity threshold δ, the period is assigned to that node and its prototype is updated. Otherwise, it initializes a provisional node. Algorithm 1 summarizes the procedure. After establishing the node topology, we map periods from trajectories to the retained nodes using the same (b_k, h_k) representation. A period-level trajectory is consequently mapped to a graph path $\bar{\tau} = (p_1, \ldots, p_K) \longrightarrow \tau^{\mathcal{G}} = (z_1, \ldots, z_K)$, and $e_k = (z_k, z_{k+1}) \in \mathcal{E}$.

2.2 From Terminal Outcomes to Progress Rewards

Having identified recurring transitions across trajectories, we next estimate their progress from cross-trajectory terminal outcome statistics. Specifically, we measure how the success rate of tra-

jectories containing each transition differs from the population success rate, and then refine this estimate to account for subsequently lost progress and execution-specific differences.

Progress from population success statistics The shared graph allows terminal outcomes to be aggregated over recurring transitions across trajectories. For each edge e_{ij}, we count the number of successful and failed trajectories in which it occurs:

$$N_{ij}^{+} = \sum_{\tau: y_\tau = 1} \mathbb{I}[e_{ij} \in \tau^{\mathcal{G}}], \qquad N_{ij}^{-} = \sum_{\tau: y_\tau = 0} \mathbb{I}[e_{ij} \in \tau^{\mathcal{G}}]. \tag{4}$$

Let p_0 denote the empirical averaged success rate. We estimate the success probability associated with e_{ij} as

$$P_{ij} = \widehat{P}(y = 1 \mid e_{ij}) = \frac{N_{ij}^{+} + 2p_0}{N_{ij}^{+} + N_{ij}^{-} + 2}, \tag{5}$$

where the prior count shrinks estimates with limited support toward the global success rate p_0.

For edges supported by at least five trajectories, we measure progress by the association between edge occurrence and eventual success:

$$r_{ij}^{\text{freq}} = \text{clip}\left(0.5(P_{ij} - p_0), -0.1, 0.1\right). \tag{6}$$

Thus, a transition receives positive credit when trajectories containing it succeed more frequently than average, and negative credit when its occurrence is associated with a lower success rate.

Penalization for lost progress Although edge-level success association identifies transitions that are generally predictive of task success, rewarding each edge independently can over-credit trajectories that reach a promising situation but subsequently lose this progress. We therefore penalize decreases in success probability along the graph path, encouraging the agent not only to reach promising situations but also to maintain or improve its estimated progress thereafter. Let P_{k-1} and $P_k = P_{e_k}$ denote the estimated success probabilities of the previous and current transitions. The reference starts at p_0 and a decrease in success likelihood incurs a degradation penalty

$$r_k^{\text{deg}} = \text{clip}\left(0.5 \max(0, P_{k-1} - P_k), 0, 0.1\right), \tag{7}$$

yielding the unadjusted graph progress score $r_k^{\text{graph}} = r_k^{\text{freq}} - r_k^{\text{deg}}$.

Execution-specific refinement The graph provides shared progress supervision for recurring transitions between aligned situations, but does not distinguish how each transition is executed at the low level. The same recurring transition can therefore be realized through different low-level action sequences, some of which advance the task more effectively than others while receiving the same graph reward. We therefore treat the edge success probability P_{ij} as a shared cross-trajectory prior and refine it using the visual and action dynamics of each particular transition occurrence. For $e_k = (z_k, z_{k+1})$, the residual uses the target period p_{k+1}. Its context consists of local visual and action dynamics within that period. The sampled MineCLIP features define seven temporal intervals. For each sampling interval m, we construct a visual transition token $x_m^v = \left[f^{m+1} - f^m; \|f^{m+1} - f^m\|_2\right] \in \mathbb{R}^{513}, m = 1, \ldots, 7$. In parallel, we summarize the low-level actions executed within each interval by their average action distribution $x_m^a \in \mathbb{R}^{12}$, yielding $X^v = (x_1^v, \ldots, x_7^v) \in \mathbb{R}^{7\times 513}$ and $X^a = (x_1^a, \ldots, x_7^a) \in \mathbb{R}^{7\times 12}$.

The visual and action transition sequences are independently encoded by two lightweight branches, and each consists of a linear projection, LayerNorm, GELU, and a GRU followed by a scalar prediction head. The resulting visual and action residuals, $\rho_{v,k}$ and $\rho_{a,k}$, are bounded to $[-0.5, 0.5]$. The residual branches refine the graph-derived success probability in logit space:

$$\widetilde{P}_{\tau,k} = \sigma\left(\text{logit}(P_{e_k}) + \rho_{v,k} + \rho_{a,k}\right). \tag{8}$$

The execution-specific correction is then defined by the change from the shared edge prior:

$$r_k^{\text{res}} = \text{clip}\left(0.25(\widetilde{P}_{\tau,k} - P_{e_k}), -0.025, 0.025\right), \tag{9}$$

for occurrence k in trajectory τ. The residual model is supervised by the trajectory outcome with details in Appendix A.2. The final progress reward is computed by:

$$r_k^{\text{raw}} = r_k^{\text{freq}} - r_k^{\text{deg}} + r_k^{\text{res}}. \tag{10}$$

To make this progress reward available during imagined rollouts to optimize action policy, we train a progress-reward predictor using the final progress rewards as supervision (details in Appendix A.3).

2.3 EXPLORATION WHEN PROGRESS STALLS

Progress rewards encourage recurring transitions, but do not explicitly specify a new direction when interaction repeatedly fails to make progress. We therefore monitor the agent progress and leverage the agent's spatial memory for targeted exploration.

Accumulating spatial search evidence To guide exploration, we maintain an episode-local spatial memory that records which regions have been observed, visited, or encountered as blocked during movement. Let v_t denote the task-conditioned visual relevance score provided by Li et al. (2025), which we use consistently for different instantiations. Using the agent's position and viewing direction, we approximate the spatial region covered by the current observation and associate v_t with the corresponding cells. For each cell c, we aggregate observations from distinct viewpoints, where a viewpoint is defined by the agent's position and viewing direction. Repeated observations from approximately the same viewpoint are treated as a single source of evidence, preventing stationary or repetitive behavior from artificially increasing confidence. Let V_c denote the number of distinct viewpoints from which c has been observed, and let N_c denote the number of these viewpoints whose latest visual score is below a threshold θ_v. The ratio N_c/V_c measures how consistently the cell is judged unpromising across viewpoints, while V_c measures the amount of supporting evidence. A cell is considered confidently negative only when both exceed their respective thresholds.

We define the negative-evidence ratio C_c and an exploration bonus U_c as

$$C_c^- = \frac{N_c}{\max(1, V_c)}, \qquad U_c = \begin{cases} 1, & V_c = 0, \\ 1 - C_c^-, & V_c > 0. \end{cases} \tag{11}$$

A cell is considered confidently negative only when both its source count and negative-evidence ratio meet their respective thresholds. The bonus favors unobserved cells and observed cells with a low negative-evidence ratio.

Short-range exploration When progress stagnates, the controller temporarily redirects the agent toward a nearby alternative direction. For each candidate movement direction d, let $x_d = x_t + \ell\, u(d)$ denote the endpoint of a short move from the current position x_t, where $u(d)$ is the unit vector associated with direction d and ℓ is the exploration step length. We denote by $c(d)$ the spatial cell containing x_d. Candidate directions whose endpoint regions are known to be blocked are excluded, while regions with confident negative evidence are deprioritized. The remaining candidates are ranked using

$$J_{\text{search}}(d) = \lambda_U U_{c(d)} - \lambda_N C_{c(d)}^- - \lambda_R R(d), \tag{12}$$

where $U_{c(d)}$ favors under-explored regions and $C_{c(d)}^-$ penalizes regions that are consistently unpromising across viewpoints. The revisit penalty is defined as

$$R(d) = n_{\text{visit}}(c(d)) + n_{\text{explore}}(x_d), \tag{13}$$

where $n_{\text{visit}}(c(d))$ measures how often the endpoint cell has been visited, and $n_{\text{explore}}(x_d)$ measures whether nearby positions have already been used as exploration targets. Thus, Directions that are both weakly explored and not repeatedly revisited are favored. The selected direction is followed for a short movement, after which the spatial search memory is updated and the exploration is replanned.

3 EXPERIMENTS

In this section, we evaluate PROSE on long-horizon embodied tasks through three questions: (**Q1**) whether it improves task completion and efficiency, (**Q2**) how its individual components contribute to performance, and (**Q3**) how it affects agent behavior in representative long-horizon failure cases.

Benchmark and Tasks We evaluate our method on MineDojo (Fan et al., 2022), a large-scale open-world benchmark built upon Minecraft. Following Li et al. (2025), we consider five long-horizon tasks that require exploration and interaction over extended trajectories: (1) Harvest Log in Plains (Cut a Tree), (2) Harvest Water with Bucket (Obtain Water), (3) Harvest Sand (Obtain Sand), (4) Shear Sheep (Obtain Wool), and (5) Mine Iron Ore (Mine Iron). We evaluate task performance using success rate and completion steps. Higher success rates indicate better task completion, and fewer completion steps indicate more efficient behavior. Because MineDojo evaluation is stochastic

Table 1: Success rates (%) on five MineDojo tasks. * denotes results obtained by retraining the method from scratch using its official implementation and evaluating over three random seeds. Best results are shown in bold, and the second best in underline.

Model	Cut a tree	Obtain water	Obtain sand	Obtain wool	Mine iron
VPT (Baker et al., 2022)	6.97	0.61	12.99	1.94	0.00
STEVE-1 (Lifshitz et al., 2023)	57.00	6.00	37.00	3.00	0.00
DreamerV3 (Hafner et al., 2023)	53.33	55.72	59.88	25.13	16.79
LS-Imagine (Li et al., 2025)	88.33±3.21	75.67±3.93	57.33±4.69	56.67±4.43	10.66±1.51
CPP (Li et al., 2026b)	90.00±2.65	80.33±3.79	60.00±4.58	**69.33±6.04**	12.00±1.83
DreamerV3* (Hafner et al., 2023)	63.50 ± 4.04	63.75 ± 4.43	53.75 ± 2.50	20.61 ± 5.57	16.50 ± 5.45
DreamerV3* +Ours	83.00 ± 4.69	68.00 ± 3.46	62.25 ± 3.40	25.75 ± 5.56	**25.00 ± 4.32**
LS-Imagine* (Li et al., 2025)	81.33 ± 2.73	67.00 ± 4.97	54.75 ± 7.14	60.75 ± 8.10	7.63 ± 1.60
LS-Imagine* + Ours	**92.33 ± 4.16**	**83.50 ± 3.32**	**64.00 ± 1.73**	63.33 ± 7.64	13.25 ± 1.71

across environment instances and initial conditions, we report mean and standard deviation across three different evaluation seeds.

Implementation Details We implement PROSE on top of two world-model agents, LS-Imagine (Li et al., 2025) and DreamerV3 (Hafner et al., 2023). Unless otherwise specified, the ablation and qualitative analyses use the LS-Imagine-based implementation. We use Qwen3-VL-8B-Instruct (Bai et al., 2025) to provide b_k in Sec. 2.1. The visual and action residual branches use GRUs with hidden width 32 and are trained with AdamW at a learning rate of 3×10^{-4}. Residual fine-tuning is scheduled every 100 completed trajectories. Our reproduced LS-Imagine and DreamerV3 results differ from the published numbers due in part to stochastic MineDojo initialization and sampled DreamerV3 actions during evaluation. We therefore use the reproduced results as controlled baselines for PROSE while also reporting the published numbers from Li et al. (2026b).

Our key takeaways are as follows:

• **(A1) PROSE improves task completion and learning efficiency across different world-model agents:** Tables 1 and 2 compare PROSE with previous methods on five long-horizon MineDojo tasks. Integrating PROSE with both LS-Imagine and DreamerV3 consistently improves task success and completion efficiency. With LS-Imagine, for example, success on *Cut a tree* increases from 81.33% to 92.33%, while steps to first success decrease from 445.52 to 323.42. With DreamerV3, the same task improves from 63.50% to 83.00%, with steps decreasing from 628.50 to 405.01. Across the ten controlled backbone–task settings, PROSE improves the point estimates of both success rate and steps to first success in every case, indicating that the gains are not confined to a particular task or base agent.

The magnitude of the gains varies across tasks, reflecting challenges beyond progress estimation alone. On *Obtain wool*, the gains are more modest but remain consistent across both agents: success improves from 60.75% to 63.33% with LS-Imagine and from 20.61% to 25.75% with DreamerV3. This task additionally requires fine-grained interaction with a dynamic target, which PROSE does not explicitly model. On *Mine iron*, PROSE improves LS-Imagine from 7.63% to 13.25%, while integrating PROSE with the stronger DreamerV3 baseline further improves success from 16.50% to 25.00%. This suggests that the weaker Mine iron result with LS-Imagine is partly due to the limitations of the base agent, while PROSE remains effective when integrated with a stronger backbone. Further diagnostic analysis is provided in Appendix C.

• **(A2) The components of PROSE provide complementary benefits:** In Table 3, the progress graph improves the success rate from 81.33% to 86.25%, showing that cross-trajectory outcome statistics provide useful intermediate supervision. However, it does not reduce the steps to first success, because the shared graph reward assigns the same progress estimate to different low-level realizations of the same recurring transition and therefore does not explicitly favor more efficient executions. Penalizing lost progress provides an additional gain, while the execution-specific refinement substantially reduces the steps from 438.81 to 340.43 by distinguishing different visual-action realizations. Targeted exploration strategy can further improve the performances.

• **(A3) PROSE recovers better from failure cases:** Fig. 3 compares LS-Imagine and PROSE across several representative long-horizon failures. The baseline may stall near a relevant target

Table 2: Average steps to first success on five MineDojo tasks. * denotes results obtained by retraining the method from scratch using its official implementation and evaluating over three random seeds. Best results are shown in bold, and the second best in underline.

Model	Cut a tree	Obtain water	Obtain sand	Obtain wool	Mine iron
VPT (Baker et al., 2022)	963.32	987.65	880.54	987.49	–
STEVE-1 (Lifshitz et al., 2023)	752.47	989.07	770.40	992.36	–
DreamerV3 (Hafner et al., 2023)	711.22	628.79	548.76	841.14	1789.06
LS-Imagine (Li et al., 2025)	363.27±28.45	473.05±35.62	612.27±42.18	558.04±31.27	1874.07±127.35
CPP (Li et al., 2026b)	<u>340.48±22.13</u>	<u>455.81±29.84</u>	567.84±38.76	**456.45±39.52**	1867.53±115.68
DreamerV3* (Hafner et al., 2023)	628.50 ± 39.74	607.91 ± 18.93	596.70 ± 22.66	896.28 ± 51.25	<u>1776.95 ± 76.06</u>
DreamerV3*+Ours	405.01 ± 39.69	524.95 ± 30.34	<u>558.83 ± 34.14</u>	832.00 ± 41.55	**1670.74 ± 55.35**
LS-Imagine* (Li et al., 2025)	445.52 ± 18.07	567.57 ± 35.33	616.43 ± 49.29	511.02 ± 63.36	1901.72 ± 23.22
LS-Imagine* +Ours	**323.42 ± 24.14**	**401.74 ± 17.47**	**540.21 ± 20.91**	<u>490.69 ± 59.50</u>	1858.89 ± 41.19

Table 3: Ablation study on the *Cut a tree* task (mean ± std over 3 runs).

Method	Success rates (%) ↑	Average steps to first success ↓
Baseline	81.33 ± 2.73	445.52 ± 18.07
+ Progress graph	86.25 ± 6.18	451.60 ± 35.36
+ Penalization for lost progress	87.33 ± 6.8	438.81 ± 31.84
+ Execution-specific refinement	91.25 ± 4.35	340.43 ±39.32
+ Exploration when stalled	92.33 ± 4.16	323.42 ± 24.14

without completing the task or become trapped in locally constrained regions. In contrast, PROSE more often converts promising states into successful completion and recovers from blocked or ineffective trajectories. These examples qualitatively illustrate improved interaction completion and recovery under long-horizon failures.

4 Related Work

Open-World Embodied Reinforcement Learning Open-world reinforcement learning is challenging due to large and evolving state spaces, long interaction horizons, and sparse task feedback. MBRL methods improve behavior learning in this setting by predicting environment dynamics and optimizing policies through imagined trajectories (Hafner et al., 2019; Yuan et al., 2026; Maes et al., 2026; Zhang et al., 2026b). For long-horizon open-world tasks, LS-Imagine (Li et al., 2025) combines short-term imagination with goal-conditioned jumpy transitions to expose longer-range consequences, while Li et al. (2026b) adapts exploration across task phases using decomposition, memory, and uncertainty-aware prediction. Beyond world models, prior work leverages demonstrations (Escoriza et al., 2025), large-scale behavior pretraining or instruction-conditioned control (Baker et al., 2022; Lifshitz et al., 2023; Zhou et al., 2024a), and explicit task structures refined from experience (Lee et al., 2026; 2022; Kim et al., 2021). PROSE instead focuses on using accumulated trajectory outcomes to infer intermediate progress under sparse terminal rewards.

Temporal Credit Assignment Temporal credit assignment studies how delayed trajectory-level outcomes should be attributed to intermediate decisions. Previous approaches redistribute delayed rewards using predicted returns, aligned demonstrations, or dynamics-informed state importance (Arjona-Medina et al., 2019; Patil et al., 2020; Verma & Metcalf, 2024). Recent work also leverages large language models for temporal credit assignment, either by directly transforming sparse outcomes into denser credit signals (Chen et al., 2026; Qu et al., 2025) or by exploiting hierarchical plan–execution structures in language agents (Feng et al., 2026; He et al., 2026; Peng et al., 2026). Other methods infer credit by estimating how intermediate decisions influence later outcomes for each trajectory, using hindsight evaluation or transition dynamics to identify behaviorally consequential states and actions (Tan et al., 2026; Liu et al., 2026). PROSE instead estimates intermediate progress by aggregating terminal outcomes over recurring transitions across trajectories.

Hierarchical Reinforcement Learning Temporal abstraction has long been used in reinforcement learning to organize primitive interactions into temporally extended behaviors (Sutton et al., 1999). Building on this idea, hierarchical reinforcement learning (HRL) reduces long-horizon de-



Figure 3: Qualitative comparison between LS-Imagine and PROSE across different tasks. PROSE better sustains and recovers task progress across diverse interaction failures.

cision making through intermediate subgoals, landmarks, or graph-based abstractions. Prior work discovers subgoals or landmarks from experience (Paul et al., 2019; Huang et al., 2019; Kim et al., 2021; Lee et al., 2022), while recent methods further generate, refine, or replan subgoals using trajectory structure, reachability, or hierarchical planning (Baek et al., 2025; Li et al., 2026a; Han et al., 2026; Haramati et al., 2026; Zeng et al., 2026). PROSE instead uses trajectory-derived structure to aggregate terminal outcomes over recurring transitions and estimate intermediate progress for policy learning, without treating recurring situations as explicit subgoals.

5 Conclusion and Future Directions

We introduced PROSE for learning intermediate progress from sparse terminal outcomes in long-horizon embodied tasks. PROSE takes a retrospective, cross-trajectory approach by identifying recurring behavior patterns across completed successful and failed trajectories and using their outcome associations as intermediate supervision. It further complements the shared cross-trajectory estimate with trajectory-specific cues that account for subsequently lost progress and differences in how the same recurring transition is executed, while targeted exploration redirects behavior when progress stalls. Across five MineDojo tasks, integrating PROSE with two world-model agents consistently improves task success and completion efficiency, with an average 9.1-point increase in success rate and a 14.4% reduction in steps to first success across agent–task settings. These results suggest that cross-trajectory outcome supervision can complement different world-model agents by turning sparse terminal feedback into useful intermediate progress signals.

Several directions remain for future work. More sample-efficient progress estimation and stronger execution recovery could improve robustness in challenging regimes, particularly when successful trajectories are scarce or task completion depends on fine-grained interaction. The gains observed across different world-model agents motivate extending PROSE to a broader range of model-based RL systems and embodied tasks. Another important direction is scaling cross-trajectory progress learning to larger experience pools without continually increasing graph complexity.

AI USE STATEMENT

In this work, we used generative AI tools to aid and polish writing, assist with retrieval and discovery of related work, and support research execution, including code revision and debugging. We did not use generative AI to generate synthetic datasets or prove mathematical claims, and the remaining disclosure categories are not applicable to this work. All AI-assisted work was reviewed by the authors. References suggested through AI-assisted literature search were verified against the original papers before inclusion. AI-assisted code modifications were manually inspected, tested, and validated by the authors, and AI-assisted writing was critically reviewed and revised before inclusion in the manuscript. The research questions, method design, experimental protocol, analysis, and scientific conclusions were determined by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the key implementation and evaluation details required to reproduce PROSE in the main paper and appendix. Section 2 describes the trajectory abstraction, cross-trajectory graph construction, progress estimation, execution-specific refinement, and targeted exploration. Section 3 reports the benchmark tasks, evaluation metrics, baseline setup, and main implementation settings, while the appendix provides additional details on temporal segmentation, residual learning, progress-reward prediction, optimization, and online updates. We report results over multiple random seeds and provide ablation studies for the major components of the method. Upon acceptance, we will release the source code, configuration files, and evaluation scripts needed to reproduce the reported experiments.

REFERENCES

Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

Seungho Baek, Taegeon Park, Jongchan Park, Seungjun Oh, and Yusung Kim. Graph-assisted stitching for offline hierarchical reinforcement learning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 2391–2408. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/baek25a.html.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Bowen Baker, Ilge Akkaya, Peter Zhokov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654, 2022.

Wen-Tse Chen, Jiayu Chen, Fahim Tajwar, Hao Zhu, Xintong Duan, Ruslan Salakhutdinov, and Jeff Schneider. Retrospective in-context learning for temporal credit assignment with large language models. *Advances in Neural Information Processing Systems*, 38:71973–71998, 2026.

Adrià López Escoriza, Nicklas Hansen, Stone Tao, Tongzhou Mu, and Hao Su. Multi-stage manipulation with demonstration-augmented reward, policy, and world model learning. *arXiv preprint arXiv:2503.01837*, 2025.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *Advances in Neural Information Processing Systems*, 35: 18343–18362, 2022.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *Advances in Neural Information Processing Systems*, 38:46375–46408, 2026.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3):440, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025.

Seungyul Han, Jaebak Hwang, Sanghyeon Lee, and Jeongmo Kim. Strict subgoal execution: Reliable long-horizon planning in hierarchical reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 146336–146361, 2026.

Dan Haramati, Carl Qi, Tal Daniel, Amy Zhang, Aviv Tamar, and George D Konidaris. Hierarchical entity-centric reinforcement learning with factored subgoal diffusion. In *International Conference on Learning Representations*, volume 2026, pp. 137484–137510, 2026.

Shuo He, Lang Feng, Xin Cheng, Lei Feng, Bo An, et al. Hierarchy-of-groups policy optimization for long-horizon agentic tasks. In *International Conference on Learning Representations*, volume 2026, pp. 27572–27593, 2026.

Zhiao Huang, Fangchen Liu, and Hao Su. Mapping state space using landmarks for universal goal reaching. *Advances in Neural Information Processing Systems*, 32, 2019.

Junsu Kim, Younggyo Seo, and Jinwoo Shin. Landmark-guided subgoal generation in hierarchical reinforcement learning. *Advances in neural information processing systems*, 34:28336–28349, 2021.

Seungjae Lee, Jigang Kim, Inkyu Jang, and H Jin Kim. Dhrl: A graph-based approach for long-horizon and sparse hierarchical reinforcement learning. *Advances in Neural Information Processing Systems*, 35:13668–13678, 2022.

Seungjoon Lee, Suhwan Kim, Minhyeon Oh, Youngsik Yoon, and Jungseul Ok. Experience-based knowledge correction for robust planning in minecraft. In *International Conference on Learning Representations*, volume 2026, pp. 137856–137908, 2026.

Jiajian Li, Qi Wang, Yunbo Wang, Xin Jin, Yang Li, Wenjun Zeng, and Xiaokang Yang. Open-world reinforcement learning over long short-term imagination. In *International Conference on Learning Representations*, volume 2025, pp. 30010–30032, 2025.

Yunfan Li, Eric Liu, and Lin Yang. Trajectory graph learning: Aligning with long trajectories in reinforcement learning without reward design. *Advances in Neural Information Processing Systems*, 38:143909–143935, 2026a.

Zhuoling Li, Xiaogang Xu, Zhenhua Xu, SerNam Lim, and Hengshuang Zhao. Larm: Large auto-regressive model for long-horizon embodied intelligence. *arXiv preprint arXiv:2405.17424*, 2024.

Ziyu Li, Chenghua Duan, YuTian Liu, Zhuo Wang, Xiuxing Li, Qing Li, Xia Wu, et al. Cognitive predictive processing: A human-inspired framework for adaptive exploration in open-world reinforcement learning. *Advances in Neural Information Processing Systems*, 38:83918–83945, 2026b.

Shalev Lifshitz, Keiran Paster, Harris Chan, Jimmy Ba, and Sheila McIlraith. Steve-1: A generative model for text-to-behavior in minecraft. *Advances in Neural Information Processing Systems*, 36: 69900–69929, 2023.

Yuze Liu, Chaochao Lu, and Chao Yang. Shadow: Dynamic-aware credit assignment against long-horizon tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 23935–23944, 2026.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworld-model: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.

Vihang P Patil, Markus Hofmarcher, Marius-Constantin Dinu, Matthias Dorfer, Patrick M Blies, Johannes Brandstetter, Jose A Arjona-Medina, and Sepp Hochreiter. Align-rudder: Learning from few demonstrations by reward redistribution. *arXiv preprint arXiv:2009.14108*, 2020.

Sujoy Paul, Jeroen Vanbaar, and Amit Roy-Chowdhury. Learning from trajectories via subgoal discovery. *Advances in Neural Information Processing Systems*, 32, 2019.

Jiangweizhi Peng, Yuanxin Liu, Ruida Zhou, Charles Fleming, Zhaoran Wang, Alfredo Garcia, and Mingyi Hong. Hiper: Hierarchical reinforcement learning with explicit credit assignment for large language model agents. *arXiv preprint arXiv:2602.16165*, 2026.

Yun Qu, Yuhang Jiang, Boyuan Wang, Yixiu Mao, Cheems Wang, Chang Liu, and Xiangyang Ji. Latent reward: Llm-empowered credit assignment in episodic reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 39, pp. 20095–20103, 2025.

Junru Song, Hengzhe Jin, Yucong Huang, Tingsong Jiang, Weien Zhou, Feifei Wang, Yang Yang, Ying Wen, and Wen Yao. Model-based imaginative planning for embodied agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18125–18147, 2026.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Hui-Ze Tan, Xiao-Wen Yang, Hao Chen, Jie-Jing Shao, Yi Wen, Yuteng Shen, Weihong Luo, Xiku Du, Lan-Zhe Guo, and Yu-Feng Li. Hindsight credit assignment for long-horizon llm agents. *arXiv preprint arXiv:2603.08754*, 2026.

Mudit Verma and Katherine Metcalf. Hindsight priors for reward learning from human preferences. In *International Conference on Learning Representations*, volume 2024, pp. 57461–57479, 2024.

Lingyi Wang, Rashed Shelim, Walid Saad, and Naren Ramakrishnan. Dmwm: Dual-mind world model with long-term imagination. *Advances in Neural Information Processing Systems*, 38: 4911–4945, 2026.

Haoqi Yuan, Chi Zhang, Hongcheng Wang, Feiyang Xie, Penglin Cai, Hao Dong, and Zongqing Lu. Skill reinforcement learning and planning for open-world long-horizon tasks. *arXiv preprint arXiv:2303.16563*, 2023.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026.

Xianghua Zeng, Hao Peng, Yicheng Pan, Angsheng Li, and Guanlin Wu. Structural information-based hierarchical diffusion for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 38:109539–109567, 2026.

Jiahan Zhang, Muqing Jiang, Nanru Dai, Taiming Lu, Arda Uzunoglu, Shunchi Zhang, Yana Wei, Jiahao Wang, Vishal Patel, Paul Liang, et al. World-in-world: World models in a closed-loop world. In *International Conference on Learning Representations*, volume 2026, pp. 55660–55699, 2026a.

Wancong Zhang, Basile Terver, Artem Zholus, Soham Chitnis, Harsh Sutaria, Mido Assran, Randall Balestriero, Amir Bar, Adrien Bardes, Yann LeCun, et al. Hierarchical planning with latent world models. *arXiv preprint arXiv:2604.03208*, 2026b.

Enshen Zhou, Yiran Qin, Zhenfei Yin, Yuzhou Huang, Ruimao Zhang, Lu Sheng, Yu Qiao, and Jing Shao. Minedreamer: Learning to follow instructions via chain-of-imagination for simulated-world control. *arXiv preprint arXiv:2403.12037*, 2024a.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024b.

A METHODOLOGY DETAILS

This section complements the main method description in Sec. 2 with additional methodological details of PROSE. Appendix A.1 details the temporal segmentation used to obtain visually coherent periods, Appendix A.2 describes how the execution-specific residual module is trained from trajectory outcomes, and Appendix A.3 explains how the retrospectively computed progress rewards are distilled into a predictor for imagined behavior learning.

A.1 TEMPORAL SEGMENTATION IN TRAJECTORY ABSTRACTION

Given a trajectory

$$\tau = \left(\{(o_t, a_t)\}_{t=1}^{T}, y\right), \tag{14}$$

where o_t and a_t denote the observation and action at environment step t, respectively, and $y \in \{0, 1\}$ denotes the terminal task outcome, we subsample the visual observations every five environment steps while always retaining the final observation. This produces

$$\tilde{o}_1, \ldots, \tilde{o}_N, \qquad N \approx T/5. \tag{15}$$

Each sampled frame is encoded using a frozen MineCLIP visual encoder ϕ_{MC},

$$f_n = \phi_{\mathrm{MC}}(\tilde{o}_n). \tag{16}$$

We compute the pairwise cosine-similarity matrix

$$S_{ij} = \frac{f_i^\top f_j}{\|f_i\|_2 \|f_j\|_2}, \qquad S \in \mathbb{R}^{N \times N}. \tag{17}$$

For each candidate boundary c, let L_c and R_c denote the sampled observations in the left and right temporal windows around c, respectively. We prefer boundaries for which observations are visually coherent within each side but dissimilar across the boundary. We therefore compute

$$\mathrm{sep}(c) = \frac{\mathrm{mean}(S_{L_c L_c}) + \mathrm{mean}(S_{R_c R_c})}{2} - \mathrm{mean}(S_{L_c R_c}), \tag{18}$$

where diagonal self-similarities are excluded from the within-window means, except for singleton windows.

To suppress boundaries caused by noisy or unstable similarities, we further penalize high variance in the within-left, within-right, and cross-window similarities:

$$\mathrm{score}(c) = \mathrm{sep}(c) - 0.5 \frac{\mathrm{std}(S_{L_c L_c}) + \mathrm{std}(S_{R_c R_c}) + \mathrm{std}(S_{L_c R_c})}{3}. \tag{19}$$

We use windows of 10 sampled observations on each side of c, corresponding to approximately 50 environment steps per side. A candidate boundary is retained when its score exceeds 0.03 and it is a strict local maximum. Neighboring boundaries are required to be separated by at least three sampled positions, corresponding to approximately 15 environment steps. If no candidate satisfies these criteria, we additionally allow a local peak whose prominence ratio is at least 0.1.

These boundaries partition the original trajectory into

$$\bar{\tau} = (p_1, \ldots, p_K), \qquad K \ll T, \tag{20}$$

where

$$p_k = \{(o_t, a_t)\}_{t=\ell_k}^{u_k}, \tag{21}$$

and ℓ_k and u_k denote the starting and ending environment steps of period p_k, respectively. Each period is therefore a temporally contiguous segment with relatively coherent visual content.

A.2 Execution-Specific Residual Module Learning

As described in Sec. 2.2, each occurrence k of a recurring graph edge $e_k = (z_k, z_{k+1})$ starts from a shared cross-trajectory success probability P_{e_k} and is adjusted by a visual residual $\rho_{v,k}$ and an action residual $\rho_{a,k}$. The residual branches are trained using trajectory-level success and failure outcomes.

Let $\mathcal{D}_{\text{fit}}$ denote the trajectory set used to fit the residual module. For a graph edge $e = e_{ij}$, let $P_e = P_{ij}$ denote its shared success probability defined in Eq. (5). For each trajectory $\tau \in \mathcal{D}_{\text{fit}}$, let $y_\tau \in \{0, 1\}$ denote its terminal task outcome, $I_{\tau,e}$ the set of occurrences of edge e in τ, and E_τ the set of distinct mapped graph edges appearing in τ.

Because the same edge may occur multiple times within one trajectory, we first aggregate its occurrence-specific corrected logits:

$$\ell_{\tau,e} = \frac{1}{|I_{\tau,e}|} \sum_{k \in I_{\tau,e}} \left[\text{logit}(P_e) + \rho_{v,k} + \rho_{a,k}\right]. \tag{22}$$

This prevents repeated visits to the same recurring transition from increasing its contribution solely through frequency.

We then average across the distinct edges in the trajectory to obtain a trajectory-level success prediction:

$$\hat{y}_\tau = \sigma\left(\frac{1}{|E_\tau|} \sum_{e \in E_\tau} \ell_{\tau,e}\right). \tag{23}$$

The aggregation is fixed and monotonic, so the learned residuals adjust the shared edge priors without introducing an additional trajectory-level prediction head.

The residual module is optimized with class-balanced binary cross-entropy:

$$\mathcal{L}_{\text{res}} = -\frac{1}{|\mathcal{D}_{\text{fit}}|} \sum_{\tau \in \mathcal{D}_{\text{fit}}} w_{y_\tau} \left[y_\tau \log \hat{y}_\tau + (1 - y_\tau) \log(1 - \hat{y}_\tau)\right], \tag{24}$$

where w_1 and w_0 compensate for the imbalance between successful and failed trajectories in the fitting split. Edge priors and feature-normalization statistics are computed from the same fitting split. After training, the learned visual and action residuals are used to obtain the execution-specific correction r_k^{res} described in Sec. 2.2.

A.3 Learning a Progress-Reward Predictor

The progress rewards defined in Sec. 2.2 are computed retrospectively, after the terminal outcome of a completed trajectory becomes available. These rewards cannot be queried directly during imagined rollouts. We therefore use the retrospectively annotated trajectories to train a separate progress-reward predictor that can be evaluated from world-model latent states.

For each completed trajectory, the final progress reward r_k^{raw} associated with transition occurrence k is assigned to the corresponding environment step in replay. Let x_t denote the world-model latent feature at step t, and let g_t denote the resulting progress-reward target at that step. We use a binary mask $m_t \in \{0, 1\}$ to indicate whether step t has an assigned progress target. The original environment-reward head is retained and trained separately from the progress-reward head.

The progress predictor $p_\psi(g_t \mid x_t)$ is trained using a masked weighted negative log-likelihood,

$$\mathcal{L}_{\text{prog}} = -\frac{\sum_t m_t \omega_t \log p_\psi(g_t \mid \text{sg}(x_t))}{\max\left(1, \sum_t m_t \omega_t\right)}, \tag{25}$$

where $\text{sg}(\cdot)$ stops gradients into the world-model latent representation, and ω_t is the per-target training weight. Thus, the progress head learns from retrospectively assigned supervision without changing the learned world-model dynamics.

During imagined rollouts, the progress head can be evaluated directly on imagined latent states. The actor and critic use a mixture of the predicted environment reward and predicted progress reward,

$$\hat{r}_t^{\text{imag}} = \alpha(s)\hat{r}_t^{\text{env}} + [1 - \alpha(s)]\,\text{clip}\left(\hat{r}_t^{\text{prog}}, -c_{\text{pred}}, c_{\text{pred}}\right), \tag{26}$$

where $\hat{r}_t^{\text{env}}$ and $\hat{r}_t^{\text{prog}}$ denote the predicted environment and progress rewards, respectively, s is the training step, and $\alpha(s)$ controls their relative contribution. We initialize $\alpha(s)$ to favor the environment reward and anneal it after a warm-up period, as detailed in Appendix B.1.

This design transfers progress supervision that is available only after trajectory completion into a reward signal that can be used during imagination. It does not modify the world-model transition architecture and does not require VLM queries on imagined states.

B Additional Experimental Details

This section complements the experimental setup in Sec. 3. We first provide the implementation and optimization details used across PROSE experiments, and then describe the controlled reproduction of the LS-Imagine and DreamerV3 baselines used for controlled comparison. We also include additional ablation studies in Sec. B.3.

B.1 Additional Implementation Details

Trajectory representation and graph construction Following Sec. 2.1, each trajectory is first partitioned into visually coherent periods p_k. We use a frozen MineCLIP image encoder both for temporal segmentation and for constructing the visual-temporal period descriptor h_k. Period boundaries are obtained using the procedure in Appendix A.1.

For each period p_k, Qwen3-VL-8B-Instruct (Bai et al., 2025) produces a structured semantic description. From this output, we retain the task, event stage, target visibility, target accessibility, and view quality to form the coarse semantic descriptor $b_k = (q, s_k, v_k, c_k, w_k)$ used for graph-node retrieval. The finer descriptor h_k is constructed from the ordered MineCLIP features and the start-to-end visual change defined in Sec. 2.1. Thus, each period is represented by (b_k, h_k) before being matched to a recurring situation z_i.

Candidate graph nodes are restricted to those sharing the same coarse descriptor b_k, after which h_k is matched to the corresponding prototype μ_i by cosine similarity. The matching threshold is δ, and retained nodes must satisfy the minimum trajectory-support criterion $m = 5$, as described in Algorithm 1. Movement metadata is kept separate from node identity, and no manually annotated intermediate progress labels are used.

Execution-specific residual learning For each occurrence k of a recurring transition $e_k = (z_k, z_{k+1})$, the shared edge prior P_{e_k} is adjusted using the visual residual $\rho_{v,k}$ and action residual $\rho_{a,k}$ defined in Sec. 2.2. The target period is represented by seven temporal intervals. Each visual token $x_m^v \in \mathbb{R}^{513}$ contains the MineCLIP feature difference and its magnitude, while the corresponding action token $x_m^a \in \mathbb{R}^{12}$ contains the average low-level action distribution in that interval.

The visual and action sequences are encoded by two independent lightweight branches. Each branch consists of a linear projection, LayerNorm, GELU, a GRU, and a LayerNorm–linear scalar head. The visual and action projection widths are 64 and 32, respectively, and both GRUs use hidden width 32. The resulting residual logits correspond to $\rho_{v,k}$ and $\rho_{a,k}$.

Feature means and standard deviations are fitted on the residual fitting set $\mathcal{D}_{\text{fit}}$ defined in Appendix A.2; dimensions with standard deviation below 10^{-5} use unit scale. Action intervals include actions following the left sampled observation up to and including the right sampled observation.

The offline residual module is optimized with AdamW using learning rate 3×10^{-4}, weight decay 10^{-3}, batch size 64 trajectories, and gradient-norm clipping at 5. Training runs for at most 100 epochs, with early stopping after 15 epochs without a validation ROC-AUC improvement greater than 10^{-5}. The class weights in $\mathcal{L}_{\text{res}}$ are

$$w_1 = \frac{0.5}{\pi}, \qquad w_0 = \frac{0.5}{1-\pi},$$

where π denotes the success fraction in $\mathcal{D}_{\text{fit}}$. We fit three independently initialized residual models and average their predicted residual logits when annotating progress rewards. Edge priors and feature-normalization statistics are computed on the same fitting split.

Progress-reward prediction and reward integration The final retrospective progress reward r_k^{raw} from Sec. 2.2 is converted into the per-step replay target g_t described in Appendix A.3. The progress head is trained from the world-model latent feature x_t using the supervision mask m_t, while gradients are stopped at the latent representation. The original environment-reward head is retained and trained separately.

During imagined rollouts, the actor and critic use the mixed reward $\hat{r}_t^{\text{imag}}$ defined in Appendix A.3. The mixing coefficient $\alpha(s)$ starts at 1.0, so behavior learning initially relies on the predicted environment reward $\hat{r}_t^{\text{env}}$. After a 100,000-step warm-up, $\alpha(s)$ decreases linearly to 0.05 over the following 200,000 steps. The annealing-start step is stored in the checkpoint so that resumed training preserves the same schedule. Predicted progress rewards $\hat{r}_t^{\text{prog}}$ are clipped with $c_{\text{pred}} = 1.0$ before reward mixing.

Online updates The progress graph is updated every 100 completed trajectories. New nodes are retained after satisfying the configured minimum support criterion $m = 5$, while unsupported provisional nodes expire after 500 trajectories of pending-update age. After a topology update, period-to-node assignments are recomputed before the edge outcome statistics P_{ij} and r_{ij}^{freq} are rebuilt.

Residual fine-tuning is also scheduled every 100 completed trajectories once at least 200 trajectories are available. Each update uses at most 2,000 trajectories and reserves 20% for validation, with at least five examples from each outcome class. Online residual fine-tuning uses learning rate 10^{-4} for 30 epochs.

Targeted exploration The exploration controller follows the spatial-memory formulation in Sec. 2.3. During environment interaction, the controller maintains the viewpoint count V_c and negative-evidence statistics N_c for each observed spatial cell c, from which the negative-evidence ratio C_c^- and exploration bonus U_c are computed. Previously visited and blocked regions are retained in the same episode-level memory.

When the learned progress estimate persistently stalls, the controller temporarily evaluates candidate movement directions d. Candidates terminating in known blocked regions are removed, while the remaining directions are ranked using $J_{\text{search}}(d)$ from Eq. 12, which combines the exploration bonus $U_{c(d)}$, negative evidence $C_{c(d)}^-$, and revisit penalty $R(d)$. After executing a short movement, the spatial memory is updated and candidate directions are recomputed if exploration continues. Control returns to the learned actor after the targeted exploration phase terminates.

B.2 Controlled Reproduction of Baseline World-Model Agents

We retrain both LS-Imagine (Li et al., 2025) and DreamerV3 (Hafner et al., 2023) using their respective official implementations and report the reproduced results as LS-Imagine* and DreamerV3* in Tables 1 and 2. For each backbone, PROSE is evaluated using the corresponding retrained implementation and evaluation pipeline, so the reproduced baselines provide controlled references for measuring the effect of our method. The reproduced performances differ from the published results on several tasks. We therefore report both published and reproduced numbers: the published results provide reference points to the original methods, while LS-Imagine* and DreamerV3* provide controlled comparisons under the implementations and evaluation settings used for PROSE. Differences between the reproduced and published results can reflect training variation as well as evaluation stochasticity. In MineDojo, each evaluation trajectory is initialized with a different environment instance. DreamerV3 additionally samples actions from its learned policy during evaluation, which introduces further stochasticity into the reported results.

B.3 Additional Ablation Studies

Table 4 shows that the component-wise trend observed on *Cut a tree* also holds on *Obtain water*. Cross-trajectory progress supervision gives a substantial initial improvement, while the degradation penalty and execution-specific refinement provide further gains, particularly in completion efficiency. Targeted exploration yields the best final performance. Together with Table 3, this suggests that the components contribute complementary benefits across tasks.

Table 4: Ablation study on the *Obtain water* task (mean ± std over 3 runs).

Method	Success rates (%) ↑	Average steps to first success ↓
Baseline	67.00 ± 4.97	567.57 ± 35.33
+ Progress graph	77.00 ± 4.36	479.29 ± 22.74
+ Penalization for lost progress	80.00 ± 1.73	455.15 ± 19.70
+ Execution-specific refinement	82.33 ± 3.79	412.06 ± 11.96
+ Exploration when stalled	83.50 ± 3.32	401.74 ± 17.47

Table 5: Positive-labeled experience in the archived offline graph fitting pools. These statistics characterize the training experience available for progress estimation and should not be interpreted as final evaluation success rates.

Task	Positive / Train	Positive fraction
Cut a tree	2276 / 2975	76.50%
Obtain water	1326 / 2520	52.62%
Obtain sand	1034 / 2217	46.64%
Obtain wool	929 / 2146	43.29%
Mine iron	204 / 1652	12.35%

C Diagnostic Analysis and Limitations

PROSE improves task completion and learning efficiency across the evaluated MineDojo tasks. To better understand where further gains may be possible, we analyze two challenging regimes observed in our experiments and examine how PROSE behaves under them.

Limited backbone capability and progress evidence in Mine iron With LS-Imagine, PROSE improves success rate from 7.63% to 13.25% and reduces steps to first success from 1901.72 to 1858.89. However, the absolute performance remains lower than PROSE combined with DreamerV3, which is consistent with differences in the capabilities of the underlying agents. In particular, LS-Imagine relies on a goal-conditioned affordance mechanism grounded in visual observations to identify promising regions. Because iron is typically underground and cannot be directly localized from surface views, this signal provides limited guidance for discovering the target resource. PROSE improves progress supervision on top of the collected experience, but does not alter this underlying perception and affordance mechanism.

The amount of successful experience available for progress estimation also differs substantially across settings. Table 5 reports the positive-labeled trajectories in the archived LS-Imagine fitting pools. For *Mine iron*, only 204 of 1,652 trajectories are positive (12.35%), compared with 43.29–76.50% for the other tasks. Under the DreamerV3-based setting, the corresponding successful-trajectory fraction increases to 18.11% for *Mine iron*, providing richer positive evidence for cross-trajectory progress estimation.

Following the LS-Imagine training configuration, we use `fast_reset=5`, such that the same environment instance can be reused for multiple consecutive trajectories before a full reset. Environment changes caused by earlier interactions may therefore persist across training trajectories, whereas evaluation uses a fresh reset for each episode. The fractions in Table 5 consequently characterize the training experience available for progress estimation rather than final policy success rates.

Importantly, PROSE also improves the stronger retrained DreamerV3 baseline on *Mine iron*, increasing success from 16.50% to 25.00% and reducing steps to first success from 1776.95 to 1670.74. Together, these results show that PROSE remains beneficial across substantially different base-agent conditions. They also suggest that stronger underlying perception and richer successful experience can provide more favorable conditions for cross-trajectory progress learning, although their individual effects are not isolated here.

Fine-grained execution and recovery The quantitative and qualitative results show that PROSE improves progress learning and recovery across a range of settings. To identify opportunities for further improvement, Fig. 4 examines representative unsuccessful PROSE trajectories from our pilot evaluation. In the first example, the agent encounters the target sheep but fails to complete the re-

(A) Target encounter without task completion



$t = 0$	$t = 200$	$t = 600$	$t = 820$	$t = 890$	$t = 910$	$t = 930$	$t = 970$	$t = 1000$
Episode start	In tall grass	At pit edge	Facing bank	Sheep ahead	Near sheep	Target offscreen	Beside bank	No task reward

(B) Prolonged confinement by terrain



$t = 0$	$t = 100$	$t = 300$	$t = 500$	$t = 550$	$t = 650$	$t = 750$	$t = 900$	$t = 1000$
Episode start	At pit edge	Facing wall	At pit edge	Inside pit	Facing dirt	Still in pit	Looking up	No task reward

(C) Repeated return to a water-side depression



$t = 0$	$t = 100$	$t = 300$	$t = 400$	$t = 500$	$t = 600$	$t = 800$	$t = 900$	$t = 1000$
Episode start	At riverbank	By the water	Below bank	Facing wall	Above bank	By the water	Back at edge	No task reward

Figure 4: Remaining failures on *Obtain wool* in a PROSE pilot evaluation. Each row shows nine observations from one unsuccessful trajectory. (A) A sheep is visible at $t = 890$–910, but the episode ends without task completion. (B) Repeated movement near a dirt depression does not produce sustained escape. (C) The agent moves around a water-side depression and later returns to it. All three episodes end at the 1,000-step limit with zero task reward. Times are saved observation indices; intervals between selected frames vary. These selected cases do not estimate failure frequencies.

quired interaction before the episode ends. In the second, the agent remains confined by local terrain for a prolonged period despite repeated movement attempts. In the third, the agent leaves a water-side depression but later returns to another locally constrained region and again fails to complete the task. These cases suggest that some remaining errors arise beyond progress estimation itself, particularly in fine-grained interaction control and sustained recovery under difficult local geometry. PROSE already improves progress guidance and stalled exploration, while tighter integration with closed-loop interaction control and longer-term recovery memory provides a natural direction for further improvement.