VIDEO2STL: GROUNDING VLM-GENERATED TEMPORAL SPECIFICATIONS FOR ROBOT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Video-based policy learning is particularly promising, as it illustrates target behaviors without requiring action annotations or embodiment-matched demonstrations. A central challenge, however, is deciding what information should be transferred from the video to the robot. Existing approaches commonly convert visual observations into scalar similarity or value signals, or ask foundation models to directly generate reward code. While effective, these approaches can make the temporal structure of a task difficult to inspect, ground, and reuse. We present Video2STL, a framework that converts observation-only videos into parametric Signal Temporal Logic (STL) specifications and uses the resulting formal representation for robot learning. A vision-language model first extracts an embodiment-independent semantic event trace and then constructs a bank of symbolic temporal specifications. The model determines the task structure, while numerical predicate thresholds and temporal bounds are grounded from successful robot trajectories. For policy learning, we separate short- and long-timescale temporal information: short-horizon specifications provide dense rewards through rolling-window quantitative robustness, while a causal monitor over a retained long-horizon specification provides one-time progress rewards for valid temporal prefixes. The same representation is intended to support cross-embodiment transfer from human or animal videos to robot control while remaining interpretable at every stage. Across four manipulation tasks, Video2STL achieves 85.8% average success-once and 67.0% success-at-end, compared with $81.5\%/59.5\%$ for native dense PPO and $65.0\%/42.3\%$ for Text2Reward; in quadruped locomotion, the Qwen-3.8 and GPT-5.6-based Video2STL policies achieve 100% success across velocities from 0.3 to $2.1\,\mathrm{m/s}$ while remaining competitive in high-speed energy efficiency. Project webpage: video2stl.

1 INTRODUCTION

Reinforcement learning has become a practical tool for robot control, but reward design remains one of the parts that is least reusable across tasks. A reward that works well often combines geometric errors, contact heuristics, success bonuses, regularizers, and task-specific schedules. The resulting function may train a strong policy while still being difficult to interpret or modify. This problem becomes more pronounced when the desired behavior is easier to show than to describe numerically. Human and animal videos provide abundant examples of manipulation and locomotion, yet they do not expose the demonstrator's actions in the robot action space, and the morphology of the demonstrator may be very different from that of the target robot.

Recent work has moved toward more natural interfaces for reward and behavior specification along two largely parallel lines. On the visual side, learning from human video has been framed as a cross-domain imitation problem (Li et al., 2022). VIP derives dense, goal-conditioned rewards from a representation pretrained on large-scale human video (Ma et al., 2023), and contrastive vision-language models such as CLIP have been used directly as zero-shot, language-specified reward models (Rocamonde et al., 2024). Generative Value Learning moves further toward general progress estimation, using a VLM to infer per-frame task progress on over 300 real-world tasks spanning many robot embodiments, with human videos usable as in-context examples (Ma et al., 2025). On the language side, language models have served as proxy reward functions that judge behavior against a user's examples or description (Kwon et al., 2023). Text2Reward and Eureka showed that they

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.

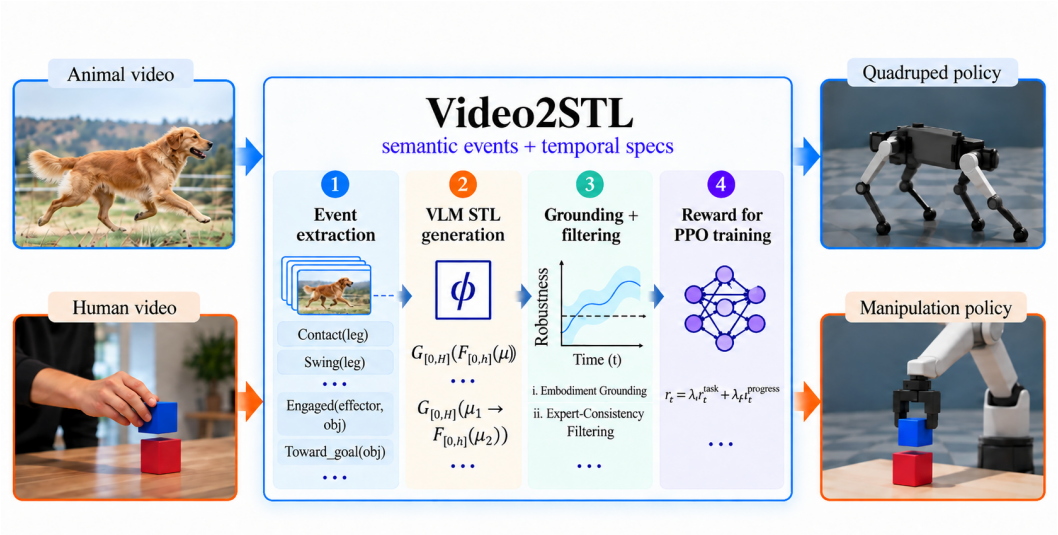


Figure 1: Overview of Video2STL framework. A VLM parses an observation video into a semantic event trace and generates parametric STL formulas. Predicate thresholds and temporal bounds are then fitted to target-embodiment trajectories. Smooth rolling-window robustness over short-horizon formulas provides dense local guidance, while a causal prefix monitor over the long-horizon backbone yields milestone progress rewards for RL policy training.

can also write dense reward programs and refine them through feedback or evolutionary search (Xie et al., 2024; Ma et al., 2024). ROSETTA constructs code-based rewards from unconstrained, evolving language preferences (Srivastava et al., 2026). Taken together, these results make a strong case that foundation models can provide task-level supervision. That supervision, however, takes the form of a scalar reward or progress signal for a task specified in language or as a goal image, which leaves open a question: what intermediate representation should connect an observation-only video to low-level robot learning?

A scalar reward is convenient for optimization, but it hides how the task unfolds over time. Consider a simple placement behavior. "Near the object", "move it toward the destination", "place it", and "leave it stable" are not four unrelated scores; they form a temporal structure. A reward program generated directly from text or video can encode this structure implicitly, but the structure is then mixed with numerical thresholds, simulator APIs, and implementation choices. Visual embedding rewards have the opposite advantage: they require little explicit structure, but it can be difficult to tell why a rollout receives a particular score or whether a high score corresponds to the intended temporal behavior. These limitations matter most in cross-embodiment settings, where pixel or pose correspondence is unreliable and where the transferable content is often semantic rather than kinematic.

Our central design choice is to separate *what the video should decide* from *what robot data should decide*. The video determines symbolic task structure: which entities matter, which semantic events occur, and how those events are ordered. Robot trajectories determine embodiment-specific numerical quantities: how close counts as *Near*, how much object motion counts as *Displaced*, and how long a temporal relation should be allowed to take. Signal Temporal Logic (STL) provides a natural interface between these two levels because it combines human-readable temporal operators with quantitative robustness semantics over real-valued robot signals (Maler & Nickovic, 2004; Fainekos & Pappas, 2009; Donzé & Maler, 2010). The same formula can therefore be read as a requirement, evaluated as a monitor, and converted into a dense learning signal.

We introduce Video2STL, a video to formal specification pipeline for robot learning. Given an observation-only video V, a vision-language model (VLM) first maps the video to an embodiment-independent semantic event trace E. A second stage maps E to a bank of parametric STL formulas Φ. We deliberately do not ask the VLM to estimate robot-specific distances, velocities, or timing constants. Instead, we collect successful robot trajectories and split them once into a grounding set and a held-out filtering set. The grounding split is used to fit predicate thresholds and temporal bounds. The filtering split is used only to test whether each grounded formula is compatible with successful target-embodiment behavior.

The second design problem is how to turn a temporal specification into an RL signal without destroying credit assignment. Prior work has shown that STL can define non-Markovian rewards and encode sequential dependencies (Venkataraman et al., 2020; Puranic et al., 2021). An earlier locomotion work also showed that the robustness of finite-history STL can provide a structured reward once temporal templates are available (Atasever et al., 2026). In our setting, however, the formulas are produced from video and can span very different timescales. We therefore use a two-timescale construction. Short formulas are evaluated on a trailing window $[t - H_{\text{local}}, t]$ and produce a dense local reward. Long formulas are not compressed into the same local window; instead, a causal prefix monitor provides a one-time reward when a new valid stage of the temporal sequence is reached. This keeps the dense signal local while preserving the ordering and deadlines of the long task.

A final issue is smooth robustness. Standard STL robustness uses nested min/max operations. Replacing them with conventional smooth approximations can improve optimization but can also change the sign of robustness, so a formula that is false under hard STL can become positive after smoothing. We use sign-preserving Boltzmann reductions whose output has the same sign as the corresponding hard min/max. This keeps the logical satisfaction boundary exact while still providing smoother within-region magnitudes for reward construction.

Our contributions are:

1. We formulate observation-only video as a source of *formal temporal task structure*. A two-stage VLM pipeline maps video to a semantic trace and then to a bank of parametric STL specifications, using one reusable ontology across manipulation tasks rather than task-specific reward templates.
2. We separate symbolic inference from embodiment-specific grounding. Numerical predicate thresholds and temporal bounds are fit on one set of successful robot trajectories, while a disjoint held-out set is used for expert-consistency filtering. The expert data calibrates and validates the VLM specifications but does not supply action supervision or define their symbolic structure.
3. We develop a two-timescale specification reward: rolling-window robustness from short formulas provides dense feedback, while a causal monitor of a long temporal backbone provides sparse, non-farmable progress.
4. We instantiate the framework in robot manipulation and quadruped locomotion, targeting cross-embodiment transfer in which the source demonstrator and target robot need not share morphology. The resulting representation remains inspectable from video interpretation through policy training.

2 Related Work

Learning Robot Behavior from Video. Early work on learning from observation focused largely on closing the domain gap between human demonstrations and robot execution. This was approached through context translation between demonstrator and robot observations (Liu et al., 2018), viewpoint-invariant visual representations for human-to-robot imitation (Sermanet et al., 2018), and domain-adaptive meta-learning with paired human and robot demonstrations (Yu et al., 2018). Li et al. (2022) further reduced the need for robot demonstrations during meta-training by translating human videos into robot-domain demonstrations. In these approaches, the information transferred from video is mainly represented through learned visual features, translated demonstrations, or adapted policies.

More recent methods use pretrained visual models to extract a learning signal directly from video. VIP learns a value-implicit representation from large-scale human video and constructs dense rewards from distances in the learned embedding space (Ma et al., 2023). RoboCLIP compares an agent's interaction trajectory with a video or language task specification using a pretrained video-language model and uses the resulting similarity as an episodic reward (Sontakke et al., 2023). GVL instead uses a VLM to estimate task progress by reasoning over the temporal ordering of video frames, with support for in-context examples from different tasks and embodiments (Ma et al., 2025).

Foundation Models for Reward Design. A related line of work uses language and vision-language models to construct reward functions from high-level task descriptions. Kwon et al. (2023) uses an LLM as a proxy reward function conditioned on a natural-language objective. Text2Reward generates executable dense reward programs from language instructions and environment APIs (Xie

et al., 2024), while Eureka iteratively improves LLM-generated reward code using downstream policy performance as feedback (Ma et al., 2024). Rocamonde et al. (2024) uses pretrained vision-language similarity directly as an RL reward, and RL-VLM-F queries a VLM for pairwise preferences over visual observations and trains a reward model from those comparisons (Wang et al., 2024). Video2Reward extracts motion keypoint trajectories from the video, prompts an LLM to generate executable reward code for legged robot learning, and iteratively refines the reward using visual feedback from the learned policy (Zeng et al., 2024). More recently, ROSETTA uses language preferences to construct staged reward code for manipulation (Srivastava et al., 2026).

Temporal Logic for Robot Learning. Prior work has explored logic-guided RL by converting temporal-logic objectives into quantitative rewards (Hasanbeig et al., 2020; Li et al., 2017; Kapoor et al., 2020; Puranic et al., 2021). Other work has studied learning temporal-logic specifications from demonstrations (Venkataraman et al., 2020; Puranic et al., 2021). Formal specifications have also guided planning and control for legged robots, including locomotion over cluttered terrain and logic-driven gait learning for quadrupeds Gu et al. (2025); Humphreys & Zhou (2025); DeFazio et al. (2024); Atasever et al. (2026).

A related line of work translates natural-language commands into formal specifications. Lang2LTL grounds complex commands into LTL for long-horizon robot navigation in previously unseen environments (Liu et al., 2023). The important distinction is the source and use of the specification. Lang2LTL starts from language and uses logic primarily to represent commands for planning/execution. In contrast, Video2STL does not assume that the temporal task structure is given in advance: the structure is extracted from video by VLM and then checked against embodiment-specific robot data.

The closest prior lines can be summarized by what they transfer from the source signal. Video imitation methods transfer appearance or behavior representations; visual-reward methods transfer scalar similarity or progress; LLM reward-generation methods transfer language intent into executable code; language-to-logic methods transfer explicit commands into symbolic temporal constraints. Video2STL focuses on a different interface between these directions: *video → explicit temporal specification → grounded quantitative reward and causal monitor*. The proposed representation is deliberately more constrained than free-form reward code and more structured than a scalar visual reward. The temporal specifications determine which quantities must be grounded, provide explicit satisfaction boundaries that can be checked on held-out expert trajectories, and support different credit-assignment mechanisms for short- and long-horizon temporal structure.

Algorithm 1: VIDEO2STL

Input: Video V, successful trajectories $\mathcal{D}^+$
Output: Policy π_θ

$E \leftarrow \mathrm{VLM}_{\mathrm{event}}(V)$
$\Phi \leftarrow \mathrm{VLM}_{\mathrm{STL}}(E)$
Split $\mathcal{D}^+ \rightarrow (\mathcal{D}_{\mathrm{ground}}, \mathcal{D}_{\mathrm{filter}})$
$\vartheta \leftarrow \mathrm{Ground}(\Phi, \mathcal{D}_{\mathrm{ground}})$
$\Phi_{\mathrm{keep}} \leftarrow \mathrm{Filter}(\Phi_\vartheta, \mathcal{D}_{\mathrm{filter}})$
Construct r_t^{local}
Construct r_t^{progress}
$r_t \leftarrow \lambda_L r_t^{\mathrm{local}} + \lambda_P r_t^{\mathrm{progress}}$
$\pi_\theta \leftarrow \mathrm{RL}(r_t)$

3 Technical Background and Preliminaries

3.1 Problem Setting

We consider a target robot controlled by a policy $\pi_\theta(a_t \mid o_t)$ in an episodic Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, T)$. We assume a source video V showing a successful behavior, and a set of successful robot trajectories $\mathcal{D}^+ = \{\tau^{(n)}\}_{n=1}^N$, where $\tau^{(n)} = (x_0^{(n)}, \ldots, x_{T_n}^{(n)})$. The source video and robot trajectories do not need to come from the same embodiment. We first prompt the VLM to extract a semantic event trace E from the source video, identifying the task-relevant events and their temporal relationships. The VLM is then prompted a second time to convert this semantic description into a bank of parametric STL formulas Φ. At this stage, the formulas define the symbolic structure of the task but leave embodiment-specific quantities, such as geometric thresholds and temporal bounds, unspecified. Successful robot trajectories are subsequently used to ground these numerical parameters and to perform expert-consistency filtering, removing formulas that are not supported by held-out successful behavior. The method follows one principle throughout: the video determines the symbolic task structure, while robot data determines its numerical parameters. This separation allows

a human or animal video to define the task without requiring morphological correspondence between the demonstrator and the robot.

3.2 Signal Temporal Logic (STL) & Quantitative Robustness

Signal Temporal Logic (STL) specifies temporal properties over real-valued signals $x(t)$ using Boolean and temporal operators (Maler & Nickovic, 2004). Formulas are built from atomic predicates $f(x(t)) \geq 0$, Boolean connectives ($\wedge, \vee, \neg$), and time-bounded temporal operators $\mathbf{G}_{[a,b]}$ (always) and $\mathbf{F}_{[a,b]}$ (eventually). We use quantitative semantics of STL (Donzé & Maler, 2010; Fainekos & Pappas, 2009); for a given trace x, and each formula φ $\rho^{\varphi}(t)$ assigns a real value capturing the degree of satisfaction. Positive values imply satisfaction; negative imply violation, and the magnitude measures the satisfaction margin. For atomic predicates, $\rho(f(x) \geq 0, x, t) = f(x(t))$. For conjunction/disjunction, robustness uses min/max: $\rho^{\varphi \wedge \psi}(t) = \min(\rho^{\varphi}(t), \rho^{\psi}(t))$, and $\rho^{\varphi \vee \psi}(t) = \max(\rho^{\varphi}(t), \rho^{\psi}(t))$. For time-bounded operators, we have $\rho^{\mathbf{G}_I \varphi}(t) = \inf_{t' \in t+I} \rho^{\varphi}(t')$, and $\rho^{\mathbf{F}_I \varphi}(t) = \sup_{t' \in t+I} \rho^{\varphi}(t')$ for an arbitrary time interval $I = [a, b]$. Parametric Signal Temporal Logic (PSTL) extends STL by allowing constants in predicates and intervals in temporal operators to be represented by parameters (Asarin et al., 2011). The *specification mining* problem seeks to infer parameter ranges where the template formula is satisfied providing a data-driven way to instantiate temporal specifications from expert-provided locomotion trajectories.

4 Methodology

4.1 Video to an Embodiment-Independent Semantic Trace

Directly asking a VLM to write a reward from pixels entangles perception, task interpretation, robot geometry, and reward engineering in one call. We instead first ask for a semantic trace that will later be translated into temporal logic. All videos use the same global ontology for manipulation and quadruped locomotion separately. For manipulation, the current entity roles are
$\mathcal{R} = \{\texttt{effector}, \texttt{object}, \texttt{receptacle}, \texttt{handle}, \texttt{articulated_part}, \texttt{goal}, \texttt{support}\}$,
and the predicate vocabulary is

$$\mathcal{P} = \{\texttt{Near}(a,b),\ \texttt{Engaged}(e,x),\ \texttt{Grasped}(o),\ \texttt{Released}(o),\ \texttt{Displaced}(o),\ \texttt{Toward_goal}(o),\ \texttt{At_goal}(o),\ \texttt{On_support}(o,s),\ \texttt{In_receptacle}(o,r),\ \texttt{Static}(x),\ \texttt{Open}(q)\}.$$

For quadruped locomotion, the current entity roles are
$\mathcal{R} = \{\texttt{trunk}, \texttt{front_left}, \texttt{front_right}, \texttt{hind_left}, \texttt{hind_right}, \texttt{support}\}$,
and the predicate vocabulary is

$$\mathcal{P} = \{\texttt{Contact}(leg),\ \texttt{Swing}(leg),\ \texttt{Touchdown}(leg),\ \texttt{Liftoff}(leg),\ \texttt{Support_count_at_least}(k, leg),\ \texttt{Trunk_height_stable}(t),\ \texttt{Trunk_pitch_stable}(t),\ \texttt{Trunk_roll_stable}(t),\ \texttt{Periodic_leg_motion}(leg)\}.$$

The first query outputs a set of event records $e_k = \big(p_k,\ c_k\big)$ where $p_k \in \mathcal{P}$ is the predicate and c_k is an event class (essential/terminal/incidental candidate). The output also includes pairwise temporal relations, such as *before*, *overlaps*, and *alternates_with*, persistent properties and a missing concepts field. If a visually important concept is absent from the ontology, the VLM must report the missing concept rather than invent a new predicate. This turns ontology insufficiency into an observable failure mode instead of silently changing the specification language.

4.2 Semantic Trace to a Parametric STL Bank

In this stage, the VLM receives the semantic trace as a JSON file, not the original video. Its purpose is to formalize temporal relationships. The model constructs a variable-size bank $\Phi = \{\phi_1, \ldots, \phi_K\}$ using only predicates supported by the trace. Numerical predicate thresholds and time constants are forbidden at this stage, only symbolic parameters such as H_{task}, h_{goal}, or h_{settle} are allowed. The current grammar contains atomic predicates, conjunction, bounded eventually and always, and nested bounded responses: $\phi ::= \mu \mid \phi_1 \wedge \phi_2 \mid \mathbf{F}_{[0,h]}\phi \mid \mathbf{G}_{[0,h]}\phi$ with implication represented through standard Boolean composition when needed. The nested form $\mathbf{F}_{[0,H]}\left(\mu_1 \wedge \mathbf{F}_{[0,h_1]}\left(\mu_2 \wedge \mathbf{F}_{[0,h_2]}\mu_3\right)\right)$ expresses a bounded event sequence, while $\mathbf{F}_{[0,H]}\mathbf{G}_{[0,h]}\mu$ expresses eventual persistence. We use a bank rather than forcing a single large formula because the source video can support multiple useful requirements at different temporal scales. For example, for PushCube, the bank contains seven

formulas. Representative examples are

$$\phi_1 = \mathbf{F}_{[0,H_{\text{task}}]}\left(\texttt{Near} \wedge \mathbf{F}_{[0,h_{\text{engage}}]}\texttt{Engaged}\right), \tag{1}$$

$$\phi_2 = \mathbf{F}_{[0,H_{\text{task}}]}\left(\texttt{TowardGoal} \wedge \mathbf{F}_{[0,h_{\text{goal}}]}\texttt{AtGoal}\right), \tag{2}$$

$$\phi_5 = \mathbf{G}_{[0,H_{\text{task}}]}\texttt{OnSupport}, \tag{3}$$

$$\phi_7 = \mathbf{F}_{[0,H_{\text{task}}]}\left(\texttt{TowardGoal} \wedge \mathbf{F}_{[0,h_{\text{goal}}]}\left(\texttt{AtGoal} \wedge \mathbf{F}_{[0,h_{\text{settle}}]}\texttt{Static}\right)\right). \tag{4}$$

The remaining formulas cover engagement-to-progress, displacement/progress, and goal-to-settling relationships. The full bank is reported in the Appendix.

4.3 Target-Embodiment Grounding

The VLM output is symbolic. To evaluate it on a robot, each predicate must be mapped to a real-valued signed margin and each symbolic temporal parameter must be assigned a numerical bound. We split the successful trajectories into disjoint grounding and filtering sets, $\mathcal{D}^+ = \mathcal{D}_{\text{ground}} \cup \mathcal{D}_{\text{filter}}$, with $\mathcal{D}_{\text{ground}} \cap \mathcal{D}_{\text{filter}} = \varnothing$. Our current experiments use a $70/30$ split. All thresholds, time bounds, and reward-normalization scales are fit from $\mathcal{D}_{\text{ground}}$ only.

Each semantic predicate p is associated with a signed atomic robustness $\rho_p(x_t; \vartheta_p)$, where ϑ_p denotes the embodiment-specific numerical parameters of the predicate and $\rho_p(x_t; \vartheta_p) \geq 0$ indicates satisfaction. We estimate ϑ_p using empirical quantiles rather than a single trajectory or hand-tuned thresholds. The choice of quantile depends on the direction of the predicate. For an upper-bound condition $p(x_t) : f_p(x_t) \leq \delta_p$, we define $\rho_p(x_t) = \delta_p - f_p(x_t)$ and choose δ_p from a high quantile of the corresponding signal on successful trajectories. Conversely, for a lower-bound condition $p(x_t) : f_p(x_t) \geq \delta_p$, we use $\rho_p(x_t) = f_p(x_t) - \delta_p$ and select δ_p from a suitably low quantile of successful behavior. The use of tolerant quantiles is particularly important for predicates that appear inside temporal operators such as $G_{[a,b]}$, whose robustness is determined by a minimum over the evaluation window. A threshold fitted to an extreme sample can make the specification unnecessarily sensitive to brief deviations. We therefore use high but non-extreme quantiles for error-type quantities, such as the 95th percentile, and low quantiles for floor-type quantities. The same principle is applied to temporal parameters: event delays observed in successful trajectories are summarized with empirical quantiles to obtain embodiment-specific temporal bounds.

4.4 Expert-Consistency Filtering

VLM-generated formulas can be syntactically valid but incompatible with target-robot behavior. After grounding, we therefore evaluate each formula on the disjoint held-out set $\mathcal{D}_{\text{filter}}$. Let $r_{ij} = \rho(\phi_i, \tau_j)$ be full-trajectory robustness of grounded formula ϕ_i on held-out trajectory τ_j. We retain

$$\phi_i \in \Phi_{\text{keep}} \iff Q_\alpha(\{r_{ij} : \tau_j \in \mathcal{D}_{\text{filter}}\}) \geq 0, \tag{5}$$

with $\alpha = 0.10$ quantile in the current experiments. For PushCube task, all seven formulas pass the $Q_{0.10} \geq 0$ test. Thus, this stage acts as held-out validation rather than aggressive pruning.

4.5 Quantitative STL and Sign-Preserving Smooth Semantics

To preserve logical satisfaction while using Boltzmann-weighted reductions, for $z \in \mathbb{R}^K$, $\beta > 0$, and a nonempty index set $A \subseteq \{1, \ldots, K\}$, define $\mathrm{BMin}_\beta(z; A) = \frac{\sum_{i \in A} e^{-\beta z_i} z_i}{\sum_{j \in A} e^{-\beta z_j}}$; BMax_β uses positive exponents. When $\min_i z_i < 0$, $\mathrm{smin}^{\mathrm{sp}}_\beta(z)$ applies BMin_β only to negative coordinates; when $\min_i z_i > 0$, it uses all coordinates; otherwise it returns zero. Symmetrically, $\mathrm{smax}^{\mathrm{sp}}_\beta(z)$ uses positive coordinates when $\max_i z_i > 0$, all coordinates when $\max_i z_i < 0$, and returns zero otherwise. These operators preserve the satisfaction boundary:

$$\mathrm{smin}^{\mathrm{sp}}_\beta(z) \geq 0 \iff \min_i z_i \geq 0, \qquad \mathrm{smax}^{\mathrm{sp}}_\beta(z) \geq 0 \iff \max_i z_i \geq 0. \tag{6}$$

4.6 Two-Timescale Temporal Reward

4.6.1 Local Rolling-Window Robustness

Long-horizon STL specifications are useful for describing the complete task, but their robustness is not always suitable as a dense reward during policy learning. We therefore use short-horizon specifications to provide local feedback over the recent trajectory history. Let $\mathrm{span}(\phi)$ denote the temporal span required to evaluate a specification ϕ. Given a local horizon H_{local}, we define

$\Phi_{\text{local}} = \{\phi_i \in \Phi_{\text{keep}} : \text{span}(\phi_i) \leq H_{\text{local}}\}$. At time t, each local specification is evaluated over the trailing window $W_t = [\max(0, t - H_{\text{local}}), t]$. For each $\phi_i \in \Phi_{\text{local}}$, we compute its quantitative robustness $\rho^{\text{sp}}_{\phi_i}(W_t)$ over W_t using the sign-preserving smooth semantics. Because robustness magnitudes can differ substantially across formulas, we normalize each specification using a scale $s_i = Q_\alpha(|\rho^{\text{sp}}\phi_i(W_t)|)$, estimated over $\tau \in \mathcal{D}\text{ground}$, with an empirical quantile. The normalized robustness is $\bar{\rho}_{\phi_i}(W_t) = \tanh\Big(\rho^{\text{sp}}_{\phi_i}(W_t)/s_i\Big)$, which bounds each contribution to $[-1, 1]$ and prevents formulas with naturally larger robustness magnitudes from dominating the reward. The local STL reward is the mean normalized robustness across the retained short-horizon specifications:

$$r^{\text{local}}_t = \frac{1}{|\Phi_{\text{local}}|} \sum_{\phi_i \in \Phi_{\text{local}}} \bar{\rho}_{\phi_i}(W_t). \tag{7}$$

Each formula first retains its own temporal semantics and satisfaction boundary, after which the normalized robustness values are combined to provide dense policy feedback. Specifications whose intrinsic temporal span exceeds this horizon are excluded from the local term and handled separately through the long-horizon causal progress monitor described next.

4.6.2 Long-Horizon Causal Progress

Short-horizon robustness provides dense feedback, but it does not by itself capture the full temporal structure of a task. A long specification may encode a sequence of events that unfolds over most of an episode, and its robustness can remain uninformative until the relevant future events occur. We therefore use a separate causal progress signal for long-horizon temporal structure.

From the retained specification bank, we select a long-horizon backbone specification $\phi_{\text{backbone}} \in \Phi_{\text{keep}}$ whose temporal structure represents the main progression of the task. The backbone is monitored causally: at time t, the monitor only uses the realized trajectory prefix $x_{0:t}$ and does not access future states. We represent the backbone as an ordered sequence of K semantic stages, $p_1 \to p_2 \to \cdots \to p_K$ together with the temporal bounds inherited from the grounded STL specification. The monitor maintains the furthest valid prefix reached so far, $b_t \in \{0, 1, \ldots, K\}$, where $b_t = k$ indicates that the first k stages have been satisfied in the required order and within their corresponding temporal constraints.

The progress reward $r^{\text{progress}}_t = (b_t - b_{t-1})/K$ is issued only when the policy reaches a new valid stage. Because b_t stores the furthest valid prefix reached so far, completed stages cannot be rewarded repeatedly, and therefore $\sum_t r^{\text{progress}}_t \leq 1$. This construction provides sparse but temporally meaningful credit for completing valid task stages, while the rolling-window robustness term supplies dense local feedback. The final reward combines the two signals:

$$r_t = \lambda_L r^{\text{local}}_t + \lambda_P r^{\text{progress}}_t. \tag{8}$$

This separation between local rolling-window robustness and long-horizon causal progress is used only for the manipulation tasks. For quadruped locomotion, we use only the local rolling-window robustness, since locomotion is primarily periodic rather than sequential and does not naturally decompose into a fixed progression of task stages.

5 Experiments

We evaluate Video2STL in two robot-learning domains with different temporal structures: quadruped locomotion and object manipulation. These domains provide complementary test cases for the proposed representation. Quadruped locomotion is approximately periodic with desired behavior characterized by recurring contact and stability patterns over time. Manipulation tasks, in contrast, typically consist of a sequence of semantically distinct stages, such as approaching an object, interacting with it, reaching a goal configuration, and maintaining the resulting state. Across both domains, the high-level pipeline is unchanged. Moreover, our main comparison in both domains contrasts three approaches to reward design: (i) a task-specific hand-engineered dense reward, (ii) *Text2Reward* (implemented using GPT-5.6 as the underlying language model), which directly generates executable reward code from a natural-language task description, and (iii) *Video2STL*, which uses an explicit temporal specification as the intermediate representation (Tao et al., 2025; Caluwaerts et al., 2023; Xie et al., 2024). All methods use the same policy architecture, optimization procedure, training budget, environment configuration, and evaluation protocol; only the reward formulation is changed.

Table 1: Quadruped locomotion performance across commanded forward velocities. Survival and success are reported as percentages, and CoT denotes cost of transportation. Higher is better for survival and success; lower is better for CoT.

	Video2STL (Qwen 3.8)			Video2STL (GPT-5.6)			Text2Reward			Heuristic		
v_x (m/s)	Surv. ↑	Succ. ↑	CoT ↓	Surv. ↑	Succ. ↑	CoT ↓	Surv. ↑	Succ. ↑	CoT ↓	Surv. ↑	Succ. ↑	CoT ↓
0.3	100%	100%	2.63 ± 0.08	100%	100%	2.63 ± 0.06	100%	100%	0.91 ± 0.01	100%	100%	1.20 ± 0.00
0.5	100%	100%	1.86 ± 0.03	100%	100%	1.80 ± 0.04	100%	100%	0.80 ± 0.00	100%	100%	1.00 ± 0.00
0.7	100%	100%	1.58 ± 0.02	100%	100%	1.52 ± 0.03	100%	100%	0.75 ± 0.00	100%	100%	1.00 ± 0.00
1.0	100%	100%	1.37 ± 0.01	100%	100%	1.26 ± 0.01	100%	100%	0.74 ± 0.00	100%	100%	1.20 ± 0.00
1.3	100%	100%	1.22 ± 0.01	100%	100%	1.17 ± 0.01	100%	100%	0.79 ± 0.00	100%	100%	1.30 ± 0.00
1.6	100%	100%	0.92 ± 0.01	100%	100%	1.12 ± 0.01	100%	100%	0.87 ± 0.00	100%	100%	1.40 ± 0.00
1.9	100%	100%	0.98 ± 0.01	100%	100%	1.13 ± 0.01	100%	100%	1.00 ± 0.01	100%	100%	1.40 ± 0.00
2.0	100%	100%	1.00 ± 0.01	100%	100%	1.13 ± 0.01	100%	100%	1.07 ± 0.01	100%	5%	1.40 ± 0.00
2.1	100%	100%	1.02 ± 0.01	100%	100%	1.15 ± 0.01	100%	100%	1.14 ± 0.01	100%	0%	1.40 ± 0.00

5.1 Quadruped Locomotion

Robot and Simulation Setup. We evaluate locomotion on Google's Barkour vb quadruped in MuJoCo XLA (MJX) (Caluwaerts et al., 2023; Todorov et al., 2012). The control timestep is $0.02\,\mathrm{s}$ and the policy outputs 12 normalized joint-position commands, which are converted to actuator torques by the robot's low-level PD controller. The policy receives the commanded linear and angular velocities together with robot proprioception and the previous action. Policies are trained using the JAX-based PPO implementation provided by Brax (Schulman et al., 2017; Freeman et al., 2021).

Evaluation Metrics. We evaluate locomotion policies under a set of fixed commanded forward velocities with zero lateral and yaw commands. Each commanded velocity is evaluated using 20 independent rollouts of 500 simulation steps. The first 50 steps are treated as a warm-up period and excluded from command-tracking statistics. We report three task-level metrics. *Survival rate* measures the fraction of rollouts that reach the full evaluation horizon without triggering an environment termination condition. *Velocity-tracking success* measures whether the mean post-warm-up local forward velocity is within 15% of the commanded velocity. Specifically, for rollout k, $\bar{v}_x^{(k)} = (T-50)^{-1}\sum_{t=51}^{T} v_{x,\mathrm{loc}}^{(k)}(t)$ and the rollout is successful if $\left|\bar{v}_x^{(k)} - v_{x,\mathrm{cmd}}\right| \le 0.15|v_{x,\mathrm{cmd}}|$. We also report the cost of transportation (CoT), $\mathrm{CoT} = E/(mgd)$, where E is the total energy consumed, m is the robot mass, g is gravitational acceleration, and d is the planar distance traveled.

Results. Across commanded forward velocities from 0.3 to $2.1\,\mathrm{m/s}$, all methods achieve 100% survival, indicating that survival alone does not distinguish the learned controllers. In terms of task success, Video2STL with Qwen and with GPT-5.6 maintains 100% success across the entire velocity range, including the highest-speed commands. Text2Reward also achieves 100% success throughout the tested range. The heuristic/hand-engineered reward baseline shows a high-speed limitation, decreasing to 5% success at $2.0\,\mathrm{m/s}$ and 0% at $2.1\,\mathrm{m/s}$. These results show that the Qwen- and GPT-5.6-generated specifications yield locomotion policies that remain robust across the full commanded-speed range.

The CoT reveals a complementary trade-off in locomotion efficiency. Text2Reward achieves the lowest CoT over most low- and mid-speed commands, whereas Video2STL-Qwen becomes increasingly competitive as velocity increases. In particular, at 1.9, 2.0, and $2.1\,\mathrm{m/s}$, Video2STL-Qwen obtains CoT values of 0.98, 1.00, and 1.02, respectively, compared with 1.00, 1.07, and 1.14 for Text2Reward and 1.40 for the heuristic baseline. Video2STL-GPT-5.6 exhibits equal or lower CoT than the Qwen variant up to $1.3\,\mathrm{m/s}$ and slightly higher CoT at higher speeds, with 1.13, 1.13, and 1.15 at 1.9, 2.0, and $2.1\,\mathrm{m/s}$. Overall, both VLM-generated specification sets yield full-range task success: Qwen provides the strongest high-speed efficiency, while Text2Reward remains more energy-efficient at lower velocities. The comparison of GPT-5.6, Gemini 3.1, and Qwen configurations is reported in Appendix A.1.1, Table 2.

5.2 Robot Manipulation

Tasks and Simulation Setup. We evaluate manipulation in ManiSkill3 (Tao et al., 2025) using four tasks: PushCube-v1, StackCube-v1, LiftPegUpright-v1, and PlaceSphere-v1. Together, these tasks cover distinct forms of interaction, including planar pushing, grasp-and-stack behavior, orientation-dependent object manipulation, and object placement. All manipulation policies use state observations and are trained with the PPO implementation provided with ManiSkill.

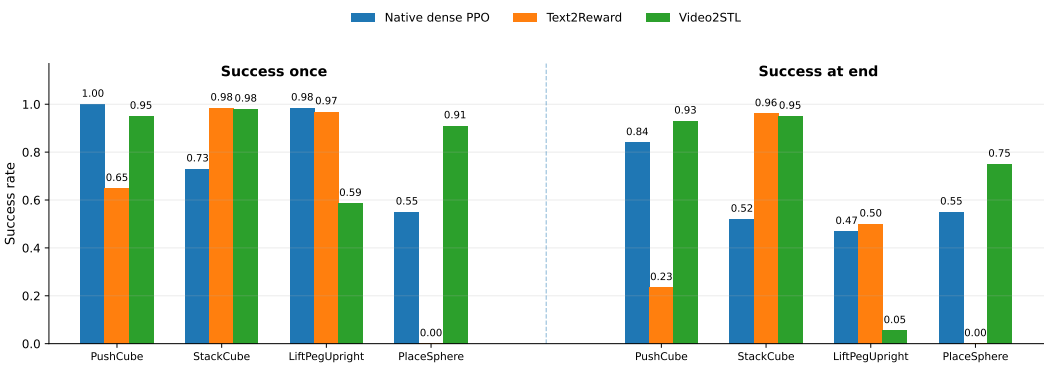


Figure 2: Manipulation results on ManiSkill3 tasks. We compare the native dense PPO reward, Text2Reward, and Video2STL on PushCube, StackCube, LiftPegUpright, and PlaceSphere.

Evaluation Metrics. We evaluate all methods using the native task success conditions provided by the corresponding ManiSkill environments. We report *success-once*, the fraction of episodes in which the native success condition is reached at least once, and *success-at-end*, the fraction of episodes satisfying the native success condition in the final state. We evaluate all policies using the same evaluation seeds and report the mean and standard deviation across 128 episodes.

Results. Averaged over the four tasks, Video2STL reaches 85.8% success-once and 67.0% success-at-end, compared with $81.5\%/59.5\%$ for native dense PPO and $65.0\%/42.3\%$ for Text2Reward (Fig. 2). On PushCube, Video2STL achieves $95\%/93\%$, retaining nearly all successes. On StackCube, it reaches $98\%/95\%$, well above native PPO ($73\%/52\%$) and on par with Text2Reward ($98\%/96\%$). The largest gain is on PlaceSphere ($91\%/75\%$ vs. $55\%/55\%$ for native PPO), which Text2Reward fails to solve. LiftPegUpright is the hardest task to retain for all methods: native PPO and Text2Reward also lose about half of their successes ($98\%/47\%$ and $97\%/50\%$). The benchmark's check only requires the peg to be upright near the table, even while still grasped, whereas the specification extracted from the demonstration additionally requires the peg to be *released* and remain static. Video2STL thus optimizes a stricter, unassisted goal: it learns the reorientation (59% success-once) but often releases the tall, narrow peg before it settles (5% success-at-end), which we trace to a grounded *Upright* tolerance ($15.9°$) looser than the peg's tipping angle ($11.8°$). Overall, grounded temporal specifications yield a learning signal competitive with native dense rewards and stronger than direct reward generation on three of four tasks.

6 Conclusion

We presented Video2STL, a framework that converts videos into interpretable temporal task specifications and uses them to guide robot learning across different embodiments. Across quadruped locomotion and four manipulation tasks, Video2STL demonstrated that video-derived temporal structure can provide effective learning signals without relying on manually engineered dense rewards. The results suggest that explicit temporal specifications provide a promising intermediate representation between visual demonstrations and robot learning, offering both behavioral interpretability and a structured mechanism for transferring task semantics across embodiments.

AI use statement

In this work, we used generative AI tools for code generation and debugging, figure generation and refinement, manuscript drafting, and LaTeX assistance. Generative AI is also part of the proposed methodology, where VLMs extract semantic event traces from videos and generate parametric STL specifications. We did not use generative AI to fabricate experimental data or quantitative results. All AI-generated code was inspected and tested, figures were manually reviewed for technical accuracy, and all manuscript text and claims were checked by the authors. We take responsibility for the final content of this work produced with the aid of generative AI.

REFERENCES

Eugene Asarin, Alexandre Donzé, Oded Maler, and Dejan Nickovic. Parametric identification of temporal properties. In *International Conference on Runtime Verification*, 2011.

Merve Atasever, Cagan Bakirci, Alfredo Reina Corona, Keyan Azbijari, and Jyotirmoy V. Deshmukh. Learning gait-aware quadruped locomotion with temporal logic specifications. *arXiv:2607.00442*, 2026.

Ken Caluwaerts, Atil Iscen, J Chase Kew, Wenhao Yu, Tingnan Zhang, Daniel Freeman, Kuang-Huei Lee, Lisa Lee, Stefano Saliceti, Vincent Zhuang, et al. Barkour: Benchmarking animal-level agility with quadruped robots. *arXiv:2305.14654*, 2023.

David DeFazio, Yohei Hayamizu, and Shiqi Zhang. Learning quadruped locomotion policies using logical rules. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 2024.

Alexandre Donzé and Oded Maler. Robust satisfaction of temporal logic over real-valued signals. In *Formal Modeling and Analysis of Timed Systems*, volume 6246 of *Lecture Notes in Computer Science*, pp. 92–106. Springer, 2010. doi: 10.1007/978-3-642-15297-9_9.

Georgios E. Fainekos and George J. Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42):4262–4291, 2009. doi: 10.1016/j.tcs.2009.06.021.

C Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax–a differentiable physics engine for large scale rigid body simulation. *arXiv:2106.13281*, 2021.

Zhaoyuan Gu, Yuntian Zhao, Yipu Chen, Rongming Guo, Jennifer K Leestma, Gregory S Sawicki, and Ye Zhao. Robust-locomotion-by-logic: Perturbation-resilient bipedal locomotion via signal temporal logic guided model predictive control. *IEEE Transactions on Robotics*, 2025.

Mohammadhosein Hasanbeig, Daniel Kroening, and Alessandro Abate. Deep reinforcement learning with temporal logics. In *International Conference on Formal Modeling and Analysis of Timed Systems*, 2020.

Joseph Humphreys and Chengxu Zhou. Learning to adapt through bio-inspired gait strategies for versatile quadruped locomotion. *Nature Machine Intelligence*, 7(7):1141–1153, 2025.

Parv Kapoor, Anand Balakrishnan, and Jyotirmoy V Deshmukh. Model-based reinforcement learning from signal temporal logic specifications. *arXiv:2011.04950*, 2020.

Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. Reward design with language models. In *International Conference on Learning Representations*, 2023.

Jiayi Li, Tao Lu, Xiaoge Cao, Yinghao Cai, and Shuo Wang. Meta-imitation learning by watching video demonstrations. In *International Conference on Learning Representations*, 2022.

Xiao Li, Cristian-Ioan Vasile, and Calin Belta. Reinforcement learning with temporal logic rewards. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017.

Jason Xinyu Liu, Ziyi Yang, Ifrah Idrees, Sam Liang, Benjamin Schornstein, Stefanie Tellex, and Ankit Shah. Grounding complex natural language commands for temporal tasks in unseen environments. In *Conference on Robotic Learning*, 2023.

YuXuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. In *IEEE International Conference on Robotics and Automation*, 2018.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. VIP: Towards universal visual reward and representation via value-implicit pre-training. In *International Conference on Learning Representations*, 2023.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. In *International Conference on Learning Representations*, 2024.

Yecheng Jason Ma, Joey Hejna, Chuyuan Fu, Dhruv Shah, Jacky Liang, Zhuo Xu, Sean Kirmani, Peng Xu, Danny Driess, Ted Xiao, Osbert Bastani, Dinesh Jayaraman, Wenhao Yu, Tingnan Zhang, Dorsa Sadigh, and Fei Xia. Vision language models are in-context value learners. In *International Conference on Learning Representations*, 2025.

Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pp. 152–166. Springer, 2004. doi: 10.1007/978-3-540-30206-3_12.

Aniruddh Puranic, Jyotirmoy Deshmukh, and Stefanos Nikolaidis. Learning from demonstrations using signal temporal logic. In *Conference on Robotic Learning*, 2021.

Juan Rocamonde, Victoriano Montesinos, Elvis Nava, Ethan Perez, and David Lindner. Vision-language models are zero-shot reward models for reinforcement learning. In *International Conference on Learning Representations*, 2024.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video. In *IEEE International Conference on Robotics and Automation*, 2018.

Sumedh A. Sontakke, Jesse Zhang, Sébastien M. R. Arnold, Karl Pertsch, Erdem Bıyık, Dorsa Sadigh, Chelsea Finn, and Laurent Itti. RoboCLIP: One demonstration is enough to learn robot policies. In *Neural Information Processing Systems*, 2023.

Sanjana Srivastava, Kangrui Wang, Yung-Chieh Chan, Tianyuan Dai, Manling Li, Ruohan Zhang, Mengdi Xu, Jiajun Wu, and Li Fei-Fei. ROSETTA: Constructing code-based reward from unconstrained language preference. In *International Conference on Learning Representations*, 2026.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse kai Chan, Yuan Gao, Xuanlin Li, Tongzhou Mu, Nan Xiao, Arnav Gurha, Viswesh Nagaswamy Rajesh, Yong Woo Choi, Yen-Ru Chen, Zhiao Huang, Roberto Calandra, Rui Chen, Shan Luo, and Hao Su. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *Robotics: Science and Systems*, 2025.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

Harish Venkataraman, Derya Aksaray, and Peter Seiler. Tractable reinforcement learning of signal temporal logic objectives. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, 2020.

Yufei Wang, Zhanyi Sun, Jesse Zhang, Zhou Xian, Erdem Biyik, David Held, and Zackory Erickson. RL-VLM-f: Reinforcement learning from vision language foundation model feedback. In *International Conference on Machine Learning*, 2024.

Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. Text2Reward: Reward shaping with language models for reinforcement learning. In *International Conference on Learning Representations*, 2024.

Tianhe Yu, Chelsea Finn, Sudeep Dasari, Annie Xie, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. In *Robotics: Science and Systems*, 2018.

Runhao Zeng, Dingjie Zhou, Qiwei Liang, Junlin Liu, Hui Li, Changxin Huang, Jianqiang Li, Xiping Hu, and Fuchun Sun. Video2reward: Generating reward function from videos for legged robot behavior learning. In *European Conference on Artificial Intelligence*, 2024.

A Appendix

A.1 Quadruped Locomotion

A.1.1 Comparison of VLM Configurations

Table 2 reports locomotion performance for the three VLM configurations. Their specifications and implementation settings are detailed below.

Table 2: Quadruped locomotion performance using STL specifications generated by different VLMs. Survival and success are reported as percentages, and CoT denotes cost of transportation. Higher is better for survival and success; lower is better for CoT.

	GPT-5.6			**Gemini 3.1**			**Qwen 3.8**		
$\mathbf{v_x}$ (m/s)	**Surv.** ↑	**Succ.** ↑	**CoT** ↓	**Surv.** ↑	**Succ.** ↑	**CoT** ↓	**Surv.** ↑	**Succ.** ↑	**CoT** ↓
0.3	100%	100%	2.63 ± 0.06	100%	25%	2.37 ± 0.09	100%	100%	2.63 ± 0.08
0.5	100%	100%	1.80 ± 0.04	100%	100%	1.97 ± 0.04	100%	100%	1.86 ± 0.03
0.7	100%	100%	1.52 ± 0.03	100%	100%	1.66 ± 0.03	100%	100%	1.58 ± 0.02
1.0	100%	100%	1.26 ± 0.01	100%	100%	1.26 ± 0.02	100%	100%	1.37 ± 0.01
1.3	100%	100%	1.17 ± 0.01	100%	100%	1.14 ± 0.01	100%	100%	1.22 ± 0.01
1.6	100%	100%	1.12 ± 0.01	100%	40%	1.16 ± 0.01	100%	100%	0.92 ± 0.01
1.9	100%	100%	1.13 ± 0.01	100%	0%	1.17 ± 0.01	100%	100%	0.98 ± 0.01
2.0	100%	100%	1.13 ± 0.01	95%	0%	1.17 ± 0.01	100%	100%	1.00 ± 0.01
2.1	100%	100%	1.15 ± 0.01	100%	0%	1.17 ± 0.01	100%	100%	1.02 ± 0.01

A.1.2 Reward Construction

For locomotion, the source observation is a video of a dog running on treadmill. (available at here) The VLM extracts embodiment-independent locomotion semantics such as limb contact, swing, touchdown and liftoff events, support structure, trunk stability, and periodic leg motion.

GPT-5.6. The semantic trace and the specification bank were produced by GPT-5.6 in a single conversation in which the video remained in context. The bank contains 11 candidates. φ_{11} relies on *Periodic_leg_motion*, a predicate that is not grounded in the reward implementation, so it was not implemented. Each of the other ten candidates was screened on rollouts of the trot expert (50 trajectories, 500 steps each, $v_x \in [0.5, 1.6]$ m/s), ignoring the first 50 steps of every trajectory, and kept only if the pooled fifth percentile of its per-step robustness was at least zero. Four candidates passed this test (φ_5, φ_6, φ_9, φ_{10}). The formulas are listed in Table 3 and the settings in Table 4.

Qwen. Qwen (`qwen3.8-max-0902`) generated stages 1–2, with the video retained in stage 2's context; GPT-5.6 generated the reward integration. Of 15 candidates, φ_{11} was rejected for using disjunction. The remaining candidates were filtered using 50 trot-expert trajectories of 500 steps, $v_x \in [0.5, 1.6]$ m/s, discarding the first 50 steps. A candidate was retained when the fifth percentile of its pooled per-step monitor outputs was non-negative, leaving seven candidates. Tables 5 and 6 give the formulas and settings.

Gemini. The Stage-2 output contained six candidate PSTL specifications (φ_1–φ_6). Parameters with matching counterparts in the supplied Barkour configuration used those values; the remaining numerical groundings retained the existing demonstration-specific settings. The candidates and the additional torque-safety and velocity-tracking specifications were screened on 50 rollouts of the supplied expert, each lasting 500 steps with $v_x \sim \mathcal{U}(0.5, 1.6)$ m/s and zero lateral and yaw commands. After discarding the first 50 steps of each rollout, a specification was retained when the pooled fifth percentile of its smooth robustness was nonnegative. Filtering acted on each complete specification. All six Gemini candidates, torque safety, and forward tracking were retained; lateral and yaw tracking were excluded.

A.1.3 Hyperparameters and Domain Randomization

We train each policy for 400M environment steps using an unroll length of 30, 32 minibatches, 4 updates per batch, discount factor $\gamma = 0.955$, learning rate 1.5×10^{-4}, entropy coefficient 0.004, 8192 parallel environments, and a batch size of 256. All reward variants use the same policy architecture and PPO optimization settings.

Table 3: GPT-5.6 specifications and final weights. Gray formulas were filtered out; φ_{11} was rejected before training. $D_\ell, L_\ell, C_\ell, S_\ell$ denote touchdown, liftoff, contact, and swing. $\mathcal{R}(p) = \mathbf{G}_{[0,H]}(\mathbf{F}_{[0,h_c]}p)$ and $\mathcal{B}(p,q) = \mathbf{G}_{[0,H]}(p \to \mathbf{F}_{[0,h_s]}q)$. Z, P denote trunk-height and pitch stability. All tanh scales are $s_i = 1$.

ID	Formula	$\mathbf{w_i}$
φ_1	$\bigwedge_{\ell\in\mathcal{L}} \mathcal{R}(D_\ell)$	0
φ_2	$\bigwedge_{\ell\in\mathcal{L}} \mathcal{R}(L_\ell)$	0
φ_3	$\mathcal{R}(D_{\mathrm{FL}} \wedge D_{\mathrm{HR}}) \wedge \mathcal{R}(D_{\mathrm{FR}} \wedge D_{\mathrm{HL}})$	0
φ_4	$\mathcal{R}(L_{\mathrm{FL}} \wedge L_{\mathrm{HR}}) \wedge \mathcal{R}(L_{\mathrm{FR}} \wedge L_{\mathrm{HL}})$	0
φ_5	$\mathcal{R}(C_{\mathrm{FL}} \wedge C_{\mathrm{HR}}) \wedge \mathcal{R}(C_{\mathrm{FR}} \wedge C_{\mathrm{HL}})$	1
φ_6	$\mathcal{R}(S_{\mathrm{FL}} \wedge S_{\mathrm{HR}}) \wedge \mathcal{R}(S_{\mathrm{FR}} \wedge S_{\mathrm{HL}})$	1
φ_7	$\mathcal{B}(D_{\mathrm{FL}}, D_{\mathrm{FR}}) \wedge \mathcal{B}(D_{\mathrm{FR}}, D_{\mathrm{FL}}) \wedge \mathcal{B}(D_{\mathrm{HL}}, D_{\mathrm{HR}}) \wedge \mathcal{B}(D_{\mathrm{HR}}, D_{\mathrm{HL}})$	0
φ_8	$\mathcal{B}(S_{\mathrm{FL}}, S_{\mathrm{HL}}) \wedge \mathcal{B}(S_{\mathrm{HL}}, S_{\mathrm{FL}}) \wedge \mathcal{B}(S_{\mathrm{FR}}, S_{\mathrm{HR}}) \wedge \mathcal{B}(S_{\mathrm{HR}}, S_{\mathrm{FR}})$	0
φ_9	$\mathbf{G}_{[0,H]}Z \wedge \mathbf{G}_{[0,H]}P$	1
φ_{10}	$\mathbf{G}_{[0,H]}(\mathit{support_count_at_least}(2))$	1
φ_{11}	$\bigwedge_{\ell\in\mathcal{L}} \mathbf{G}_{[0,H]}(\mathit{Periodic_leg_motion}(\ell))$ (rejected)	—

Table 4: GPT-5.6 grounding and reward parameters. All values are fixed in the reward implementation, except the support count, which is specified by the VLM, and the filtering rule. Triples follow walk/trot/bound mode order; $\Delta t = 0.02\,\mathrm{s}$.

Parameter	Role	Value
Temporal		
H	history-window setting, all modes	30 steps
h_c	recurrence bound	$(29, 20, 15)$ steps
h_s	response bound	$(14, 10, 7)$ steps
Mode hysteresis	walk/trot entry, exit	$0.72,\ 0.65\,\mathrm{m/s}$
	trot/bound entry, exit	$1.55,\ 1.45\,\mathrm{m/s}$
Predicates		
Contact / swing thresholds	foot clearance	$0.001,\ 0.025\,\mathrm{m}$
Contact / swing scales	robustness normalization	$0.020,\ 0.025\,\mathrm{m}$
Trunk-height tolerance	relative to nominal reset height	$0.080\,\mathrm{m}$
Trunk-pitch tolerance	angular bound	$0.30\,\mathrm{rad}$
k	minimum support count	2
Invalid-event robustness	unavailable predecessor sample	-8
Aggregation, reward, and filtering		
Smoothing temperatures	atomic / temporal / formula	$0.08,\ 0.12,\ 0.10$
s_i	specification tanh scale	1.0
Candidate aggregate	before warmup gate	$\sum_i w_i \tanh(\rho_i/s_i)/\sum_i w_i$
Specification weight	within total STL reward	1.0
$\tau_{\max}$	joint torque limit	$18\,\mathrm{N\,m}$
$w_{\mathrm{safe}}, \alpha_{\mathrm{safe}}$	safety weight and tanh scale	$0.25,\ 0.25$
γ_τ	torque-effort coefficient	10^{-6}
Reward scales	total STL / linear tracking / angular tracking	$1.0,\ 1.0,\ 0.5$
Action-rate scale	action-change penalty	-0.01
Tracking parameter	`tracking_sigma`	0.25
Filter rule	retain candidate with pooled fifth percentile	$q_{0.05} \geq 0$

We additionally apply domain randomization to friction and actuator parameters. Friction is sampled in the range $(0.6, 1.4)$, while actuator gain and bias perturbations are sampled in the range $(-5, 5)$.

A.1.4 Sources Videos & VLM Prompts For Quadruped Locomotion

We use two sequential VLM prompts for quadruped locomotion. The first extracts an embodiment-independent semantic locomotion trace directly from an animal video. The second receives only this structured trace and converts it into candidate parametric STL specifications.

Stage 1: Video to Semantic Locomotion Trace

Table 5: Qwen specifications and final weights. Gray formulas were filtered out; φ_{11} was rejected before training. $D_\ell, L_\ell, C_\ell, S_\ell$ denote touchdown, liftoff, contact, and swing. $\mathcal{R}(p) = \mathbf{G}_{[0,H]}(\mathbf{F}_{[0,h_c]}p)$ and $\mathcal{B}(p,q) = \mathbf{G}_{[0,H]}(p \to \mathbf{F}_{[0,h_s]}q)$. Z, P, R denote trunk-height, pitch, and roll stability. All tanh scales are $s_i = 1$.

ID	Formula	$\mathbf{w_i}$
φ_1	$\mathcal{R}(D_{\mathrm{HR}}) \wedge \mathcal{R}(L_{\mathrm{HR}})$	0
φ_2	$\mathcal{R}(D_{\mathrm{FR}}) \wedge \mathcal{R}(L_{\mathrm{FR}})$	0
φ_3	$\mathcal{R}(D_{\mathrm{HL}}) \wedge \mathcal{R}(L_{\mathrm{HL}})$	0
φ_4	$\mathcal{R}(D_{\mathrm{FL}}) \wedge \mathcal{R}(L_{\mathrm{FL}})$	0
φ_5	$\mathcal{B}(D_{\mathrm{HR}}, D_{\mathrm{FR}})$	1
φ_6	$\mathcal{B}(D_{\mathrm{HL}}, D_{\mathrm{FL}})$	0.70
φ_7	$\mathcal{B}(D_{\mathrm{FR}}, D_{\mathrm{FL}}) \wedge \mathcal{B}(D_{\mathrm{FL}}, D_{\mathrm{FR}})$	0
φ_8	$\mathcal{B}(D_{\mathrm{HR}}, D_{\mathrm{HL}}) \wedge \mathcal{B}(D_{\mathrm{HL}}, D_{\mathrm{HR}})$	0
φ_9	$\mathcal{R}(C_{\mathrm{HR}} \wedge C_{\mathrm{FR}})$	1
φ_{10}	$\mathcal{R}(C_{\mathrm{HL}} \wedge C_{\mathrm{FL}})$	0
φ_{11}	$\mathbf{G}_{[0,H]}C_{\mathrm{HR}} \vee \mathbf{G}_{[0,H]}C_{\mathrm{HR}}$ (rejected)	—
φ_{12}	$\mathbf{G}_{[0,H]}Z \wedge \mathbf{G}_{[0,H]}P \wedge \mathbf{G}_{[0,H]}R$	1
φ_{13}	$\mathbf{G}_{[0,H]}(\mathit{support_count_at_least}(2))$	1
φ_{14}	$\bigwedge_{\ell \in \mathcal{L}} \mathcal{R}(C_\ell)$	1
φ_{15}	$\bigwedge_{\ell \in \mathcal{L}} \mathcal{R}(S_\ell)$	1

Table 6: Qwen grounding and reward parameters, supplied by the stage-3 configuration except the VLM-specified support count and the filtering rule. Triples follow walk/trot/bound mode order; $\Delta t = 0.02\,\mathrm{s}$.

Parameter	Role	Value
Temporal		
H	history-window setting, all modes	30 steps
h_c	recurrence bound	$(29, 20, 15)$ steps
h_s	response bound	$(14, 10, 7)$ steps
Mode hysteresis	walk/trot entry, exit	$0.72,\ 0.65$ m/s
	trot/bound entry, exit	$1.55,\ 1.45$ m/s
Predicates		
Contact / swing thresholds	foot clearance	$0.001,\ 0.025$ m
Contact / swing scales	robustness normalization	$0.020,\ 0.025$ m
Trunk-height tolerance	relative to nominal reset height	0.080 m
Trunk-pitch / roll tolerances	angular bounds	$0.30,\ 0.30$ rad
k	minimum support count	2
Invalid-event robustness	unavailable predecessor sample	-8
Aggregation, reward, and filtering		
Smoothing temperatures	atomic / temporal / formula	$0.08,\ 0.12,\ 0.10$
s_i	specification tanh scale	1.0
Candidate aggregate	before warmup gate	$\sum_i w_i \tanh(\rho_i/s_i)/\sum_i w_i$
Specification weight	within total STL reward	1.0
$\tau_{\max}$	joint torque limit	18 N m
$w_{\mathrm{safe}}, \alpha_{\mathrm{safe}}$	safety weight and tanh scale	$0.25,\ 0.25$
γ_τ	torque-effort coefficient	10^{-6}
Reward scales	total STL / linear tracking / angular tracking	$1.0,\ 1.0,\ 0.5$
Action-rate scale	action-change penalty	-0.01
Tracking parameter	`tracking_sigma`	0.25
Filter rule	retain candidate with pooled fifth percentile	$q_{0.05} \geq 0$

```
System Role:
Act as an expert Vision-Language Model specializing in Biomechanics and Robotic Locomotion.

Context & Objective:
Analyze the provided video of a real quadruped animal performing locomotion. Your goal is to
      extract an embodiment-independent semantic description of the temporal locomotion
     structure. This analysis will later be used to construct Signal Temporal Logic (STL)
```

Table 7: Gemini specifications and final aggregation priors. Gray formulas were filtered out. D_ℓ and L_ℓ denote touchdown and liftoff. $\mathcal{R}(p) = \mathbf{G}_{[0,H]}(\mathbf{F}_{[0,h_c]}p)$ and $\mathcal{B}(p,q) = \mathbf{G}_{[0,H]}(p \to \mathbf{F}_{[0,h_p]}q)$. Z and P denote trunk-height and pitch stability. The first six rows are Gemini-generated; the remaining rows are implementation-supplied safety and tracking terms.

ID	Formula	$\mathbf{w_i}$
φ_1	$\mathbf{G}_{[0,H]}Z \wedge \mathbf{G}_{[0,H]}P$	1
φ_2	$\mathbf{G}_{[0,H]}(\mathit{support_count_at_least}(2))$	1
φ_3	$\mathcal{R}(D_{\mathrm{FL}}) \wedge \mathcal{R}(D_{\mathrm{HR}})$	1
φ_4	$\mathcal{R}(L_{\mathrm{HL}})$	1
φ_5	$\mathcal{B}(D_{\mathrm{HL}}, D_{\mathrm{FL}})$	1
φ_6	$\mathcal{B}(D_{\mathrm{HR}}, D_{\mathrm{FR}})$	1
φ_7	$\mathbf{G}_{[0,H]}\left(\bigwedge_{j=1}^{12}(\lvert\tau_j\rvert \le \tau_{\max})\right)$	1
φ_8	$\mathbf{G}_{[0,W]}\left(\lvert v_x - v_x^{\mathrm{cmd}}\rvert \le \epsilon_x\right)$	1
track y	$\mathbf{G}_{[0,W]}\left(\lvert v_y - v_y^{\mathrm{cmd}}\rvert \le \epsilon_y\right)$	0
track yaw	$\mathbf{G}_{[0,W]}\left(\lvert \omega_z - \omega_z^{\mathrm{cmd}}\rvert \le \epsilon_\omega\right)$	0

Table 8: Gemini grounding and reward parameters. Parenthesized pairs follow walk/trot mode order; unpaired values are shared; $\Delta t = 0.02\,\mathrm{s}$.

Parameter	Role	Value
Temporal		
H (H_{episode})	outer always bound	$(30, 24)$ steps
h_c	recurrence bound	25 steps
h_p	phase-response bound	13 steps
W	velocity/yaw tracking bound	$(30, 24)$ steps
Mode hysteresis	trot entry / walk return	$0.72,\ 0.65\,\mathrm{m/s}$
Predicates		
z_{th}	contact clearance threshold	$0.003\,\mathrm{m}$
c	contact robustness scale	$0.002\,\mathrm{m}$
Contact / swing	opposite signed margins	C_ℓ / $-C_\ell$
Touchdown / liftoff	exact contact transitions	adjacent samples
z_{ref}	reference trunk height	torso height at reset
δ_z, s_z	height tolerance / margin scale	$0.06\,\mathrm{m}$ / $0.02\,\mathrm{m}$
$\theta_{\max}$	absolute trunk-pitch tolerance	$(8^\circ, 7^\circ)$
s_θ	pitch robustness scale	5°
k	support count (kth-largest margin)	2
ϵ_x	forward tracking tolerance	$(0.55, 0.60)\,\mathrm{m/s}$
$\epsilon_y, \epsilon_\omega$	excluded lateral/yaw tolerances	$0.05\,\mathrm{m/s}$; $0.05\,\mathrm{rad/s}$
Aggregation, reward, and filtering		
$\beta_{\mathrm{atom}}, \beta_{\mathrm{temp}}$	Boolean / temporal sharpness	$20,\ 20$
β_{group}	group Boltzmann soft-min sharpness	0.5
w_i	specification prior	1 active; 0 excluded
α_g	group tanh scale	1
$w_{\mathrm{safe}}, w_{\mathrm{track}}$	safety / tracking group weights	$1,\ 1$
w_{pattern}	gait-pattern group weight	$(1.1, 1.2)$
$\tau_{\max}, s_\tau$	torque limit / margin scale	$18\,\mathrm{N\,m}$ / $5\,\mathrm{N\,m}$
Forward multiplier	multiplier of ρ_x/ϵ_x	1
Total reward rate	sum of bounded group rewards	$\sum_g w_g \tanh(\rho_g/\alpha_g)$
Filter rule	pooled retention criterion	$q_{0.05} \ge 0$

```
    specifications for a quadruped robot with different morphology, body dimensions, and
    controllers. Therefore, you must prioritize repeated, behaviorally important temporal
    structures over exact joint angles, limb trajectories, or incidental movements.

Ontology & Permitted Vocabulary:
Use only the following terms for your analysis.

1. Body Roles:
```

```
Trunk: The main body/torso of the quadruped.
Front_left: Front-left leg/foot.
Front_right: Front-right leg/foot.
Hind_left: Hind-left leg/foot.
Hind_right: Hind-right leg/foot.
Support_surface: The surface supporting locomotion, including ground or treadmill belt.

2. States & Events:

Contact(leg): Foot is visibly in contact with the support surface.
Swing(leg): Leg is in the aerial/swing phase.
Touchdown(leg): Transition from swing to contact.
Liftoff(leg): Transition from contact to swing.
Support_count_at_least(k): At least k feet are supporting the body.
Trunk_height_stable: The trunk maintains approximately bounded vertical variation over
     repeated gait cycles.
Trunk_pitch_stable: The trunk maintains approximately bounded pitch variation over repeated
     gait cycles.
Trunk_roll_stable: The trunk maintains approximately bounded roll variation over repeated
     gait cycles.
Periodic_leg_motion(leg): The leg exhibits a repeated locomotion cycle.

3. Temporal Relations:

before
after
overlaps
alternates_with (Events repeatedly occur in alternating portions of the gait cycle.)
approximately_synchronous_with (Events repeatedly occur close together.)
phase_shifted_from (Consistent nonzero temporal offset between two legs.)

4. Observation Classes:

Essential_candidate: Visually supported, embodiment-independent temporal relationships
     critical to describing the locomotion pattern.

Persistent_candidate: Body-level properties persisting through multiple gait cycles
(e.g., stable trunk height).

Incidental: Visually present but unnecessary for formal description
(e.g., head movement, tail motion, panting, a single irregular step).

Strict Constraints & Rules:

- Do NOT guess: If left/right identity or contacts cannot be reliably determined due to
     viewpoint or occlusion, explicitly report the uncertainty.

- Do NOT infer robot-specific metrics: Exclude joint angles, torques, actions, motor states,
      joint velocities, actuator properties, or numerical thresholds.

- Treadmill physics: If the animal is on a treadmill, its body may remain stationary in
     image/world coordinates. Do NOT interpret this as zero forward velocity.

Task Requirements:
Analyze the video and determine:

1. Visible body parts.
2. Repeated stance/contact and swing behaviors.
3. Repeated touchdown and liftoff ordering.
4. Strongest inter-leg temporal relationships
   (synchronization, alternation, phase offsets).
5. Persistent body-level regularities
   (e.g., stable trunk pitch).
6. Ambiguities and occlusions.

Output Format:

Return valid JSON only, using exactly this structure.
Do not include markdown code blocks or conversational text outside the JSON.

{
  "locomotion_context": {
    "type": "overground | treadmill | uncertain",
    "confidence": 0.0,
    "evidence": "brief visual justification"
  },

  "visibility": {
    "trunk": "clear | partial | unclear",
    "front_left": "clear | partial | unclear",
```

```
    "front_right": "clear | partial | unclear",
    "hind_left": "clear | partial | unclear",
    "hind_right": "clear | partial | unclear"
  },

  "observed_events": [
    {
      "event_id": "e1",
      "predicate": "touchdown(front_left)",
      "event_class":
        "essential_candidate | persistent_candidate | incidental",
      "confidence": 0.0,
      "visual_evidence": "brief explanation"
    }
  ],

  "repeated_states": [
    {
      "predicate": "contact(front_left)",
      "confidence": 0.0,
      "visual_evidence": "brief explanation"
    }
  ],

  "inter_leg_relations": [
    {
      "relation_id": "r1",
      "leg_1": "front_left",
      "relation":
        "before | after | overlaps | alternates_with |
         approximately_synchronous_with | phase_shifted_from",
      "leg_2": "hind_right",
      "event_basis": "touchdown | liftoff | contact | swing",
      "confidence": 0.0,
      "evidence": "brief explanation"
    }
  ],

  "persistent_properties": [
    {
      "predicate":
        "Trunk_height_stable | support_count_at_least(k)",
      "confidence": 0.0,
      "evidence": "brief explanation"
    }
  ],

  "uncertainties": [
    "brief description of an ambiguity or occlusion"
  ]
}
```

Stage 2: Semantic Trace to PSTL Specifications

```
System Role:
Act as an expert Formal Methods Engineer specializing in Logic Synthesis for Robotic Control.


Context & Objective:
You are constructing Parametric Signal Temporal Logic (PSTL) specifications based on a
     semantic locomotion trace. This trace (provided as a JSON input) was extracted from a
     video of a real quadruped animal. Your objective is to translate this trace into
     embodiment-independent, candidate temporal specifications that describe the
     demonstrated repeated locomotion pattern. These specifications will eventually be
     grounded to a four-legged robot with a different morphology to generate a dense,
     continuous reward signal for reinforcement learning.

Strict Constraints (The "Zero-Hallucination" Rule):

- Trace-Dependent Reasoning: You must reason only from the supplied semantic trace.

- No Invention: Do not invent contact events, predicates, or temporal relations that are
     absent from the input data. Use a predicate only if it is explicitly supported by the
     trace.

- Symbolic Grounding: All temporal constants and numerical temporal parameters must remain
     purely symbolic (e.g., H_episode, h_cycle) for later numerical grounding during
     training.
```

```
Allowed PSTL Grammar:

You are restricted to the following exact textual grammar.
(Note: mu represents an atomic predicate; phi represents a sub-formula; H and h represent
     symbolic temporal bounds).

Atomic predicate:
mu

Conjunction:
phi_1 AND phi_2

Bounded eventuality:
F_[0,H](mu)

Bounded persistence:
G_[0,H](mu)

Repeated-event requirement:
G_[0,H](F_[0,h](mu))

Response requirement:
G_[0,H](mu_1 -> F_[0,h](mu_2))

Repeated paired-event requirement:
G_[0,H](F_[0,h](mu_1 AND mu_2))

Ordered repeated response:
G_[0,H](mu_1 -> F_[0,h1](mu_2))

Note:
Conjunctions of independently valid formulas are permitted.

Output Format:

Return valid JSON only, using exactly the structure below.
Do not include markdown code blocks, conversational text, or explanations outside the JSON.

{
  "candidate_specifications": [
    {
      "candidate_id": "phi_1",

      "formula":
        "PSTL formula using ONLY the allowed textual grammar",

      "predicates_used": [
        "instantiated_predicate_1"
      ],

      "supporting_event_ids": [
        "e1"
      ],

      "supporting_relation_ids": [
        "r1"
      ],

      "excluded_incidental_event_ids": [
        "e7"
      ],

      "symbolic_parameters": [
        "H_episode",
        "h_cycle"
      ],

      "semantic_complexity": 1,

      "rationale":
        "Brief explanation grounded purely in the input semantic trace"
    }
  ],

  "stage1_ambiguities_affecting_specification": [
    "Brief description of how an ambiguity in the input trace impacts the formula"
  ]
}
```

A.2 ROBOT MANIPULATION

A.2.1 REWARD CONSTRUCTION

This section lists the parametric Signal Temporal Logic (PSTL) specifications generated by the VLM from the semantic traces of the manipulation demonstrations. These are the raw candidate specifications produced before numerical grounding using robot expert trajectories and before expert-consistency filtering. Thus, not every specification listed below is necessarily retained in the final reward.

For compactness, we denote the manipulated object by o, the robot end-effector by e, the support surface by s, and the receptacle by r. All temporal bounds, such as H_{task} and h_{goal}, are symbolic at this stage and are subsequently grounded from expert trajectories.

PushCube-v1

For PUSHCUBE, the VLM generated the following eight candidate specifications:

$$\phi_1 : \ \mathbf{F}_{[0,H_{\text{task}}]}\left(\text{Near}(e,o) \wedge \mathbf{F}_{[0,h_{\text{engage}}]}\text{Engaged}(e,o)\right), \tag{9}$$

$$\phi_2 : \ \mathbf{F}_{[0,H_{\text{task}}]}\left(\text{Toward_goal}(o) \wedge \mathbf{F}_{[0,h_{\text{goal}}]}\text{At_goal}(o)\right), \tag{10}$$

$$\phi_3 : \ \mathbf{F}_{[0,H_{\text{task}}]}\left(\text{Engaged}(e,o) \wedge \mathbf{F}_{[0,h_{\text{progress}}]}\left(\text{Toward_goal}(o) \wedge \mathbf{F}_{[0,h_{\text{goal}}]}\text{At_goal}(o)\right)\right), \tag{11}$$

$$\phi_4 : \ \mathbf{F}_{[0,H_{\text{task}}]}\text{Displaced}(o) \wedge \mathbf{F}_{[0,H_{\text{task}}]}\text{Toward_goal}(o), \tag{12}$$

$$\phi_5 : \ \mathbf{G}_{[0,H_{\text{task}}]}\text{On_support}(o,s), \tag{13}$$

$$\phi_6 : \ \mathbf{F}_{[0,H_{\text{task}}]}\left(\text{At_goal}(o) \wedge \mathbf{F}_{[0,h_{\text{settle}}]}\text{Static}(o)\right), \tag{14}$$

$$\phi_7 : \ \mathbf{F}_{[0,H_{\text{task}}]}\left(\text{Toward_goal}(o) \wedge \mathbf{F}_{[0,h_{\text{goal}}]}\left(\text{At_goal}(o) \wedge \mathbf{F}_{[0,h_{\text{settle}}]}\text{Static}(o)\right)\right), \tag{15}$$

$$\phi_8 : \ \mathbf{F}_{[0,H_{\text{task}}]}\left(\text{Toward_goal}(o) \wedge \mathbf{F}_{[0,h_{\text{goal}}]}\text{At_goal}(o)\right) \wedge \mathbf{G}_{[0,H_{\text{task}}]}\text{On_support}(o,s). \tag{16}$$

These specifications capture approach and engagement, object displacement, goal-directed progress, goal attainment, maintenance of support, and post-goal stationarity.

StackCube-v1

For STACKCUBE, the VLM generated seven candidate specifications:

$$\phi_1 : \ \mathbf{F}_{[0,H_{\text{task}}]}\left(\text{Grasped}(o) \wedge \mathbf{F}_{[0,h_{\text{displace}}]}\text{Displaced}(o)\right), \tag{17}$$

$$\phi_2 : \ \mathbf{F}_{[0,H_{\text{task}}]}\left(\text{Grasped}(o) \wedge \mathbf{F}_{[0,h_{\text{reach}}]}\left(\text{Toward_goal}(o) \wedge \mathbf{F}_{[0,h_{\text{goal}}]}\text{At_goal}(o)\right)\right), \tag{18}$$

$$\phi_3 : \ \mathbf{G}_{[0,H_{\text{task}}]}\left(\text{Toward_goal}(o) \rightarrow \mathbf{F}_{[0,h_{\text{goal}}]}\text{At_goal}(o)\right), \tag{19}$$

$$\phi_4 : \ \mathbf{F}_{[0,H_{\text{task}}]}\left(\text{On_support}(o,s) \wedge \mathbf{F}_{[0,h_{\text{release}}]}\left(\text{Released}(o) \wedge \mathbf{F}_{[0,h_{\text{settle}}]}\text{Static}(o)\right)\right), \tag{20}$$

$$\phi_5 : \ \mathbf{F}_{[0,H_{\text{task}}]}\left(\mathbf{G}_{[0,h_{\text{support}}]}\text{On_support}(o,s)\right), \tag{21}$$

$$\phi_6 : \ \mathbf{F}_{[0,H_{\text{task}}]}\left(\mathbf{G}_{[0,h_{\text{static}}]}\text{Static}(o)\right), \tag{22}$$

$$\begin{aligned}\phi_7 : \ & \mathbf{F}_{[0,H_{\text{task}}]}\left(\text{Grasped}(o) \wedge \mathbf{F}_{[0,h_{\text{reach}}]}\left(\text{Toward_goal}(o) \wedge \mathbf{F}_{[0,h_{\text{goal}}]}\text{At_goal}(o)\right)\right) \\ & \wedge \mathbf{F}_{[0,H_{\text{task}}]}\left(\text{On_support}(o,s) \wedge \mathbf{F}_{[0,h_{\text{release}}]}\left(\text{Released}(o) \wedge \mathbf{F}_{[0,h_{\text{settle}}]}\text{Static}(o)\right)\right) \\ & \wedge \mathbf{F}_{[0,H_{\text{task}}]}\left(\mathbf{G}_{[0,h_{\text{support}}]}\text{On_support}(o,s)\right) \wedge \mathbf{F}_{[0,H_{\text{task}}]}\left(\mathbf{G}_{[0,h_{\text{static}}]}\text{Static}(o)\right).\end{aligned} \tag{23}$$

The resulting specification set describes grasping and transport toward the target, supported placement, release, and persistent terminal stability.

LiftPegUpright-v1

For LIFTPEGUPRIGHT, the VLM generated seven candidate specifications:

$$\phi_1: \quad \mathbf{F}_{[0,H_{\text{task}}]}\left(\text{Near}(e,o) \wedge \mathbf{F}_{[0,h_{\text{reach}}]}\text{Engaged}(e,o)\right), \tag{24}$$

$$\phi_2: \quad \mathbf{F}_{[0,H_{\text{task}}]}\left(\text{Grasped}(o) \wedge \mathbf{F}_{[0,h_{\text{manip}}]}\text{Displaced}(o)\right), \tag{25}$$

$$\phi_3: \quad \mathbf{F}_{[0,H_{\text{task}}]}\left(\text{Toward_goal}(o) \wedge \mathbf{F}_{[0,h_{\text{goal}}]}\text{Upright}(o)\right), \tag{26}$$

$$\phi_4: \quad \mathbf{F}_{[0,H_{\text{task}}]}\left(\text{Upright}(o) \wedge \mathbf{F}_{[0,h_{\text{terminal}}]}\left(\text{On_support}(o,s) \wedge \mathbf{F}_{[0,h_{\text{terminal2}}]}\text{At_goal}(o)\right)\right), \tag{27}$$

$$\phi_5: \quad \mathbf{F}_{[0,H_{\text{task}}]}\left(\text{Upright}(o) \wedge \mathbf{F}_{[0,h_{\text{release}}]}\left(\text{Released}(o) \wedge \mathbf{F}_{[0,h_{\text{stable}}]}\text{Static}(o)\right)\right), \tag{28}$$

$$\phi_6: \quad \mathbf{F}_{[0,H_{\text{task}}]}\left(\mathbf{G}_{[0,h_{\text{persist}}]}\text{Upright}(o)\right), \tag{29}$$

$$\phi_7: \quad \mathbf{F}_{[0,H_{\text{task}}]}\left(\mathbf{G}_{[0,h_{\text{persist2}}]}\text{On_support}(o,s)\right). \tag{30}$$

These candidates characterize approach and interaction with the peg, displacement, progress toward an upright configuration, supported placement, release followed by stability, and persistence of the terminal upright and supported states.

PlaceSphere-v1

For PLACESPHERE, the VLM generated eight candidate specifications:

$$\phi_1: \quad \mathbf{F}_{[0,H_{\text{task}}]}\left(\text{Near}(e,o) \wedge \mathbf{F}_{[0,h_{\text{reach}}]}\text{Grasped}(o)\right), \tag{31}$$

$$\phi_2: \quad \mathbf{F}_{[0,H_{\text{task}}]}\left(\text{Grasped}(o) \wedge \mathbf{F}_{[0,h_{\text{displace}}]}\left(\text{Displaced}(o) \wedge \mathbf{F}_{[0,h_{\text{progress}}]}\text{Toward_goal}(o)\right)\right), \tag{32}$$

$$\phi_3: \quad \mathbf{F}_{[0,H_{\text{task}}]}\left(\text{Toward_goal}(o) \wedge \mathbf{F}_{[0,h_{\text{place}}]}\left(\text{In_receptacle}(o,r) \wedge \mathbf{F}_{[0,h_{\text{release}}]}\text{Released}(o)\right)\right), \tag{33}$$

$$\phi_4: \quad \mathbf{G}_{[0,H_{\text{task}}]}\left(\text{Released}(o) \rightarrow \mathbf{F}_{[0,h_{\text{settle}}]}\text{Static}(o)\right), \tag{34}$$

$$\phi_5: \quad \mathbf{F}_{[0,H_{\text{task}}]}\left(\mathbf{G}_{[0,h_{\text{in_receptacle}}]}\text{In_receptacle}(o,r)\right), \tag{35}$$

$$\phi_6: \quad \mathbf{F}_{[0,H_{\text{task}}]}\left(\mathbf{G}_{[0,h_{\text{static}}]}\text{Static}(o)\right), \tag{36}$$

$$\phi_7: \quad \mathbf{F}_{[0,H_{\text{task}}]}\left(\mathbf{G}_{[0,h_{\text{in_receptacle}}]}\text{In_receptacle}(o,r)\right) \wedge \mathbf{F}_{[0,H_{\text{task}}]}\left(\mathbf{G}_{[0,h_{\text{static}}]}\text{Static}(o)\right), \tag{37}$$

$$\phi_8: \quad \mathbf{F}_{[0,H_{\text{task}}]}\left(\text{Near}(o,r) \wedge \mathbf{F}_{[0,h_{\text{enter}}]}\left(\text{In_receptacle}(o,r) \wedge \mathbf{F}_{[0,h_{\text{release}}]}\text{Released}(o)\right)\right). \tag{38}$$

The candidate set represents the demonstrated sequence of approaching and grasping the object, transporting it toward the receptacle, placing and releasing it, and maintaining containment and stationarity after placement.

Example Grounding - PushCube:

Let $p_o(t)$, $p_e(t)$, and p_g denote the positions of the object, end-effector, and goal, respectively. We compute the end-effector distance $d_{eo}(t) = \|p_e(t) - p_o(t)\|_2$, planar goal distance $d_g(t) = \|p_{o,xy}(t) - p_{g,xy}\|_2$, displacement $d_{\text{disp}}(t) = \|p_{o,xy}(t) - p_{o,xy}(0)\|_2$, cumulative goal progress $P(t) = d_g(0) - d_g(t)$, and instantaneous motion $m(t) = \|p_{o,xy}(t) - p_{o,xy}(t-1)\|_2$. The grounded atomic margins are then formulated as:

$$\begin{aligned} \rho_{\text{Near}}(t) &= \delta_{\text{near}} - d_{eo}(t), & \rho_{\text{TowardGoal}}(t) &= P(t) - \delta_{\text{progress}}, \\ \rho_{\text{Displaced}}(t) &= d_{\text{disp}}(t) - \delta_{\text{displaced}}, & \rho_{\text{AtGoal}}(t) &= \delta_{\text{goal}} - d_g(t), \\ \rho_{\text{OnSupport}}(t) &= \delta_z - |z_o(t) - z_{\text{support}}|, & \rho_{\text{Static}}(t) &= \delta_{\text{static}} - m(t), \end{aligned} \tag{39}$$

with $\rho_{\text{Engaged}}(t) = \min\{\rho_{\text{Near}}(t),\, m(t) - \delta_{\text{engaged}}\}$.

The current PushCube parameters are $\delta_{\text{near}}, \delta_{\text{engaged}}, \delta_{\text{displaced}}, \delta_{\text{progress}}, \delta_{\text{goal}}, \delta_z, \delta_{\text{static}}$ and temporal bounds are $H_{\text{task}}, h_{\text{engage}}, h_{\text{progress}}, h_{\text{goal}}, h_{\text{settle}}$.

A.2.2 HYPERPARAMETERS

For all four tasks, we use the standard ManiSkill PPO implementation without modifying the PPO optimization procedure. The common hyperparameters are: learning rate 3×10^{-4}, discount factor $\gamma = 0.8$, GAE parameter $\lambda = 0.9$, PPO clipping coefficient 0.2, value-function coefficient 0.5, target KL divergence 0.1, 8 PPO update epochs per batch, and 32 minibatches. The actor and critic are three-layer MLPs with 256 hidden units per layer and $\tanh$ activations.

For PushCube, we use 2048 parallel environments, a rollout length of 20, and a total training budget of 2M environment steps.

For StackCube, we use 1024 parallel environments, a rollout length of 50, and a total training budget of 30M environment steps.

For LiftPegUpright, we use 1024 parallel environments, a rollout length of 50, and a total training budget of 25M environment steps.

For PlaceSphere, we use 1024 parallel environments, a rollout length of 50, and a total training budget of 10M environment steps.

A.2.3 Source Videos & VLM Prompts For Manipulation

For each manipulation task, we record a single human demonstration. The demonstration is processed by the VLM using two sequential, task-agnostic prompts designed for general manipulation. The first prompt extracts a semantic event trace using the shared manipulation ontology, while the second maps this trace to a bank of parametric STL specifications. Importantly, the same prompts are applied to all manipulation tasks without any task-specific modification.

Stage 1: Human Video to Semantic Manipulation Trace

```
System Role:
Act as an expert Vision-Language Model specializing in Biomechanics, Robotic Manipulation,
     and Signal Temporal Logic (STL).

Context & Objective:
Analyze the provided video of a human performing a manipulation task. Your goal is to
     extract an embodiment-independent semantic description of the temporal manipulation
     structure. This analysis will be used to construct STL specifications for a robot with
     a different morphology, body dimensions, and controllers. Therefore, you must describe
     what happens semantically rather than reproducing the demonstrator's exact trajectory,
     human kinematics, speed, or incidental movements.

Ontology & Permitted Vocabulary:

Use only the following roles and predicates in formal semantic fields.

1. Entity Roles:

Effector: The component directly interacting with the environment (e.g., human hand, robot
     end-effector).
Object: A manipulable object whose state or position may change.
Receptacle: A container or region into which an object can be placed.
Handle: A graspable/interactable component used to manipulate an articulated object.
Articulated_part: A movable articulated component (e.g., drawer, door).
Goal: A visually identifiable destination or goal region.
Support: A surface or structure physically supporting an object.

2. States & Events (Predicates):

Near(a, b): Entity a is spatially close to entity b.
Engaged(effector, x): The effector is actively interacting with x to change its state/motion.

Grasped(object): The object is securely held by the effector.
Released(object): The object is no longer grasped by the effector.
Displaced(object): The object has moved meaningfully from its initial location.
Toward_goal(object): The object has made meaningful progress toward a visual goal.
Upright(object): The object is in an approximately upright or standing orientation.
At_goal(object): The object occupies a visually identifiable goal region.
On_support(object, support): The object remains supported by the intended supporting surface.

In_receptacle(object, receptacle): The object occupies the interior/accepted region of a
     receptacle.
Static(x): Entity x is approximately stationary.
Open(articulated_part): The articulated part is in an open configuration.

3. Temporal Relations:

before
after
overlaps
persists_until

4. Observation Classes:
```

```
Essential_candidate: Visually supported, embodiment-independent temporal relationships
     critical to describing the manipulation.
Terminal_candidate: An observed state/event that plausibly represents completion or the
     resulting outcome.
Incidental: Visually present but unnecessary for formal description.

Strict Constraints & Rules:

1. A role, state, or event may be absent from the video.
  Do not force every role or state to be populated.

2. Do NOT infer joint angles, velocities, or numerical thresholds.

3. Do NOT use human-specific variables
  (e.g., hand_x, wrist_angle, elbow_pose)
  inside semantic predicates.

4. Do NOT interpret exact human timing for the future robot.

Task Requirements:

1. Identify the visually relevant entities in the video and map them to the permitted
     semantic roles.

2. Formulate the temporal relationships among observed events
  using ONLY the permitted predicates.

3. Identify persistent properties only when the video gives clear evidence that a property
     is maintained for a meaningful interval.

4. Note any critical ambiguities or visual occlusions.

Output Format:

Return valid JSON ONLY. Do not include markdown formatting,
code blocks, or conversational text outside the JSON structure.
Use exactly this schema:

{
 "manipulation_context": {
  "confidence": 0.0,
  "evidence":
    "Brief visual justification of the overall task being performed"
 },

 "entity_mapping": {
  "effector":
    "Description of what serves as the effector (or null)",
  "object":
    "Description of the object (or null)",
  "receptacle":
    "Description of the receptacle (or null)",
  "handle":
    "Description of the handle (or null)",
  "articulated_part":
    "Description of the articulated part (or null)",
  "goal":
    "Description of the goal region (or null)",
  "support":
    "Description of the support surface (or null)"
 },

 "visibility": {
  "effector": "clear | partial | unclear | n/a",
  "object": "clear | partial | unclear | n/a",
  "receptacle": "clear | partial | unclear | n/a",
  "handle": "clear | partial | unclear | n/a",
  "articulated_part": "clear | partial | unclear | n/a",
  "goal": "clear | partial | unclear | n/a",
  "support": "clear | partial | unclear | n/a"
 },

 "observed_events": [
  {
   "event_id": "e1",
   "state_or_event": "e.g., Grasped(object)",
   "event_class":
     "essential_candidate | terminal_candidate | incidental",
   "confidence": 0.0,
   "visual_evidence":
     "Brief explanation of supporting visual evidence"
```

```
    }
  ],

  "temporal_relations": [
    {
      "event_1": "e1",
      "relation": "before | after | overlaps | persists_until",
      "event_2": "e2",
      "confidence": 0.0
    }
  ],

  "persistent_properties": [
    {
      "state_or_event":
        "e.g., On_support(object, support)",
      "confidence": 0.0,
      "visual_evidence": "Brief explanation"
    }
  ],

  "missing_concepts": [
    {
      "description":
        "Important observed concept not expressible by the strict ontology",
      "importance": "low | medium | high"
    }
  ],

  "uncertainties": [
    "Brief statement describing an important ambiguity or occlusion"
  ]
}
```

Stage 2: Semantic Manipulation Trace to PSTL Specifications

```
System Role:
Act as an expert Formal Methods Engineer specializing in Logic Synthesis for Robotic Control.


Context & Objective:
You are constructing Parametric Signal Temporal Logic (PSTL)
specifications based on a semantic manipulation trace.
This trace (provided as a JSON input) was extracted from a human demonstration video.

Your objective is to translate this trace into embodiment-independent, candidate temporal
     specifications that describe the demonstrated behavior. These specifications will
     eventually be grounded to a robot with a different morphology to generate a dense,
     continuous reward signal for reinforcement learning.

Strict Constraints:

- Trace-Dependent Reasoning: You must reason only from the supplied semantic trace.
- No Invention: Do not invent contact events, predicates, or temporal relations that are
     absent from the input data. Use a predicate only if it is explicitly supported by the
     trace.

Symbolic Grounding:

All temporal constants and numerical temporal parameters must remain purely symbolic (e.g.,
     H_task, h_persist, h_reach) for later numerical grounding during training.

Allowed PSTL Grammar:

You are restricted to the following exact textual grammar.
(Note: mu represents an atomic predicate; phi represents a sub-formula; H and h represent
     symbolic temporal bounds).

Atomic predicate:
mu

Conjunction:
phi_1 AND phi_2

Bounded eventuality:
F_[0,H](mu)

Bounded persistence:
G_[0,H](mu)
```

```
Response requirement:
G_[0,H](mu_1 -> F_[0,h](mu_2))

Two-event ordered sequence:
F_[0,H](mu_1 AND F_[0,h1](mu_2))

Three-event ordered sequence:
F_[0,H](mu_1 AND F_[0,h1](mu_2 AND F_[0,h2](mu_3)))

Terminal stability:
F_[0,H](G_[0,h](mu))

Conjunctions of independently valid formulas are permitted.

Output Format:

Return valid JSON only, using exactly the structure below.
Do not include markdown code blocks, conversational text, or
explanations outside the JSON.

{
  "ontology_sufficiency":
    "sufficient | partially_sufficient | insufficient",

  "ontology_note":
    "Brief explanation of whether the available grammar and predicates adequately captured
        the trace.",

  "candidate_specifications": [
    {
      "candidate_id": "phi_1",

      "description":
        "Plain English description of the behavior this formula enforces",

      "formula":
        "PSTL formula using exactly the allowed textual grammar",

      "predicates_used": [
        "instantiated predicate (e.g., Grasped(object))"
      ],

      "supporting_event_ids": [
        "e1",
        "e2"
      ],

      "excluded_incidental_event_ids": [
        "e4"
      ],

      "ordered_events": [
        "instantiated predicate 1",
        "instantiated predicate 2"
      ],

      "persistent_requirements": [],

      "symbolic_parameters": [
        "H_task",
        "h1"
      ],

      "semantic_complexity":
        "low | moderate | high",

      "rationale":
        "Brief explanation grounded only in the provided semantic trace"
    }
  ],

  "stage1_ambiguities_affecting_specification": [
    "Brief explanation of any ambiguity in the input trace that impacted formula construction
        "
  ]
}
```