Runtime Monitoring of Perception-Based Autonomous Systems via Embedding Temporal Logic

Parv Kapoor[*] [†]
Software and Societal Systems Department
Carnegie Mellon University
parvk@andrew.cmu.edu

Abigail Hammer[*]
Software and Societal Systems Department
Carnegie Mellon University
arhammer@andrew.cmu.edu

Ashish Kapoor
General Robotics
ashish@generalrobotics.company

Karen Leung
Aeronautics and Astronautics Department
University of Washington
kymleung@uw.edu

Eunsuk Kang
Software and Societal Systems Department
Carnegie Mellon University
eunsukk@andrew.cmu.edu

Abstract

Runtime monitoring of autonomous systems traditionally relies on mapping continuous sensor observations to discrete logical propositions defined over low-dimensional state variables. In perception-driven systems, constructing such predicates often requires additional perception modules that can be brittle, computationally expensive, and difficult to generalize across tasks. Building on prior work on Embedding Temporal Logic (ETL), we develop a runtime monitoring framework that directly evaluates perceptual and temporal properties over learned observation embeddings. Our approach constructs predicates from distances to reference observations and composes them with temporal operators to monitor sequential behaviors without requiring explicit state estimation. We formalize finite-prefix monitoring semantics for online evaluation, introduce data-driven predicate calibration procedures with a conformal recall guarantee, and characterize the computational overhead of incremental monitoring. We evaluate ETL monitors across navigation, simulated manipulation, and real-world robotic demonstrations. Our results demonstrate strong agreement with ground-truth monitors and show that temporal composition enables the detection of ordering violations that atomic similarity-based monitoring cannot reliably identify.

1 Introduction

Modern autonomous systems, from self-driving vehicles to robotic manipulators, increasingly rely on learned representations for perception, prediction, and decision making [Hafner et al., 2020, Hansen et al., 2024, Zhou et al., 2025, Kim et al., 2024, Intelligence et al., 2025, Ye et al., 2026]. These representations allow autonomous systems to overcome the challenges of operating on explicit state space representations (such as object poses and velocities), which often require state estimation pipelines

[*]Indicates equal contribution
[†]Work done during an internship at General Robotics

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

and auxiliary localization modules. We refer to systems that operate on learned representations as *perception-based systems*. Perception-based systems map high-dimensional sensor streams such as images, video, or lidar to compact latent representations, which are then consumed by downstream policies, planners, or world models [NVIDIA et al., 2026, Baniodeh et al., 2025].

A promising approach to achieving high-assurance autonomy involves formally specifying desired properties of a system and applying techniques such as formal verification [Luckcuck et al., 2019] and runtime monitoring [Maler and Nickovic, 2004] to check whether the system satisfies these properties [Seshia et al., 2018]. In particular, *runtime monitoring* has attracted considerable interest as it can be deployed *online* to provide rigorous guarantees about the system behavior without incurring the cost of an exhaustive offline analysis. A runtime monitor periodically evaluates the execution of a system and raises an alert when the system exhibits undesirable behavior [Colombo and Pace, 2022]. This ability to provide lightweight but rigorous online assurance has led to successful applications in domains such as autonomous vehicles [Schön and Lindemann, 2026], drones [Gu et al., 2023], and robotic manipulators [Lin and Baras, 2019]. Runtime monitoring relies on the availability of *formal specifications* that capture the desired properties of a system. For systems with low-dimensional state representations, specification notations such as *signal temporal logic (STL)* [Maler and Nickovic, 2004] provide an expressive formalism to specify behavioral properties over logical *predicates*. Each predicate encodes a condition over a state variable that can be evaluated as true or false at each step of execution, e.g., whether its position, velocity, or force is below or above a given threshold.

However, for perception-based systems, writing such formal specifications over learned representations remains an open challenge [Seshia et al., 2018]. For these systems, low-dimensional state representations are often unavailable or require specialized, ad-hoc perception modules. For example, translating a property such as "the robot is near the obstacle" or "the gripper is holding the object" into predicates requires either an additional classifier or detector, or a handcrafted feature extractor tailored to the task (e.g., to detect whether the concept of the gripper "holding" an object is present in the current scene) [Hekmatnejad et al., 2024]. Adding these modules can introduce new sources of brittleness, calibration errors, and domain dependence into the system. Worse, whenever the vocabulary of concepts for specification evolves (e.g., to also be able to express properties about the gripper "dropping" an object), it may be necessary to augment the existing perception modules or add new ones to support the new concepts. Overall, there is a *fundamental mismatch* between (i) the latent space over which typical perception systems operate and (ii) the low-dimensional state space over which specifications in existing temporal logic notations are expressed.

Prior work [Kapoor et al., 2025a] addressed this gap by introducing learned observation embeddings as first-class elements of formal specifications. Rather than relying on explicit state variables, perceptual properties are expressed through distances between *target embeddings*, representing desired semantic concepts, and *observation embeddings* obtained during system execution. This approach leverages the geometric structure of learned representations, where semantically similar observations tend to lie close together [Radford et al., 2021, Oquab et al., 2024]. For example, proximity to an obstacle can be expressed by a predicate that checks whether the distance between the current observation embedding and a reference embedding of the obstacle falls below a threshold. Building on the embedding predicates and temporal-logic syntax of *Embedding Temporal Logic* (ETL) introduced in Kapoor et al. [2025a], we extend this framework to runtime monitoring. We consider both general-purpose vision encoders (e.g., CLIP, DINOv2, and DINOv3) and domain-matched world-model encoders (e.g., Dreamer- and video-diffusion-based models), whose representations additionally capture task-specific information. By composing embedding predicates with temporal operators, ETL enables monitoring of complex perceptual behaviors, such as requiring a robot to retain its grasp on an object until it reaches a designated location. We evaluate both classes of representations to investigate their suitability for monitoring atomic and temporally extended properties.

Although embedding-based specifications are conceptually straightforward, translating them into reliable runtime monitors presents several challenges that remain unresolved in prior work. First, there is the question of how target embeddings are generated: they can come from a reference image, a demonstration, or a set of both, and different choices can induce meaningfully different predicates. Additionally, embeddings are learned, continuous, and model-dependent representations, and geometric proximity is not guaranteed to align perfectly with the logical distinctions required for monitoring. A central challenge, therefore, is to turn embedding-space similarity into a well-defined specification primitive: one must decide which geometric relationships correspond to predicate satisfaction, how to calibrate decision thresholds, and how these predicates are composed to create system

specifications. These issues make embedding-based specifications substantially more challenging than simply reusing learned features inside an existing monitor.

In this paper, we make the following four contributions: (i) define Embedding Temporal Structures and build on prior work Kapoor et al. [2025a] to specify perceptual behaviors directly over observations using Embedding Temporal Logic (ETL) (Section 3.1); (ii) we formally define Boolean satisfaction semantics over bounded traces, yielding explicit finite-prefix online monitors for perceptual specifications over realized execution traces (Sections 3.1 and 3.2) (iii) we propose data-driven methods for calibrating embedding predicate thresholds, making them feasible for safety-oriented monitoring (Section 4); and (iv) we evaluate ETL-based monitors across navigation and manipulation domains, showing that they can faithfully monitor atomic and sequential perceptual behaviors across diverse environments (Section 5).

2 Background and Related Work

Formal Specifications for Robotic Systems Temporal logics, such as Linear Temporal Logic (LTL), STL, and Metric Temporal Logic (MTL) have been used to formally verify complex behaviors in cyber-physical and robotic systems. These logics have been used for trajectory planning [Kress-Gazit et al., 2009, Sun et al., 2022, Leung et al., 2023], reinforcement learning [Aksaray et al., 2016, Alur et al., 2023, Aloor et al., 2023], runtime monitoring [Bartocci et al., 2018], and adaptive control [Raman et al., 2014, Belta and Sadraddini, 2019, Lindemann and Dimarogonas, 2019, Kapoor et al., 2025b]. These logics can struggle with systems that rely on ML for perception, where input data can have a variable number of objects in frame and evolving bounding boxes. Recently, Spatiotemporal Perception Logic (STPL) [Hekmatnejad et al., 2024] was introduced, which combined Timed Quality Temporal Logic [Dokhanchi et al., 2018] with spatial logic and allows quantification over objects, as well as 2D and 3D spatial reasoning. Kapoor et al. [2025a] introduced Embedding Temporal Logic (ETL), defining embedding-based predicates and temporal-logic syntax for specifying perceptual behaviors, with applications to planning. Building on this foundation, we extend ETL to runtime monitoring through systematic predicate construction and calibration, a conformal recall guarantee, and comprehensive evaluation across navigation and manipulation tasks.

Pretrained Vision Encoders Recent advances in representation learning have produced pretrained vision encoders that are sufficiently expressive to serve as general-purpose perceptual representations across a wide range of visual domains [Oquab et al., 2024]. As a result, these models provide a practical foundation for extending temporal logic beyond low-dimensional state space representations. Pretrained vision encoders such as CLIP [Radford et al., 2021] and DINOv2 [Oquab et al., 2024] provide representations in which perceptual similarity can be measured, making them suitable for constructing embedding-based predicates. Alternatively, embeddings can be obtained from domain-matched world models, such as Dreamer [Hafner et al., 2025] or video models trained on task-specific data [Guo et al., 2026], whose latent representations additionally capture task dynamics. In this work, we evaluate both classes of encoders.

Specification Based Runtime Monitoring Given a logical specification that is well-defined over bounded traces and encodes a desired system property, a runtime monitor is an online evaluation at each time step during an execution to assess whether the execution satisfies the given specification [Maler and Nickovic, 2004, Bartocci et al., 2018]. In practice, runtime monitoring often goes beyond Boolean verdicts and employs quantitative semantics, such as *robustness* measures in temporal logic, which provide a real-valued signal indicating how much a trace satisfies or violates the specification [Fainekos and Pappas, 2009]. These quantitative monitors are particularly useful in continuous and stochastic systems, as they enable graded feedback and can be integrated with optimization or control algorithms for real-time decision making.

Conformal Prediction Conformal prediction [Vovk et al., 2005] is a distribution-free calibration framework that turns a held-out calibration set into a finite-sample statistical guarantee under only an *exchangeability* assumption. The exchangeability of data means that the calibration and test samples are identically distributed and order-independent. At a high level, it is a method for calibrating a model so that its predictions on unseen examples come with a reliability guarantee. We employ conformal prediction theory for calibrating thresholds for ETL predicates in Section 4.

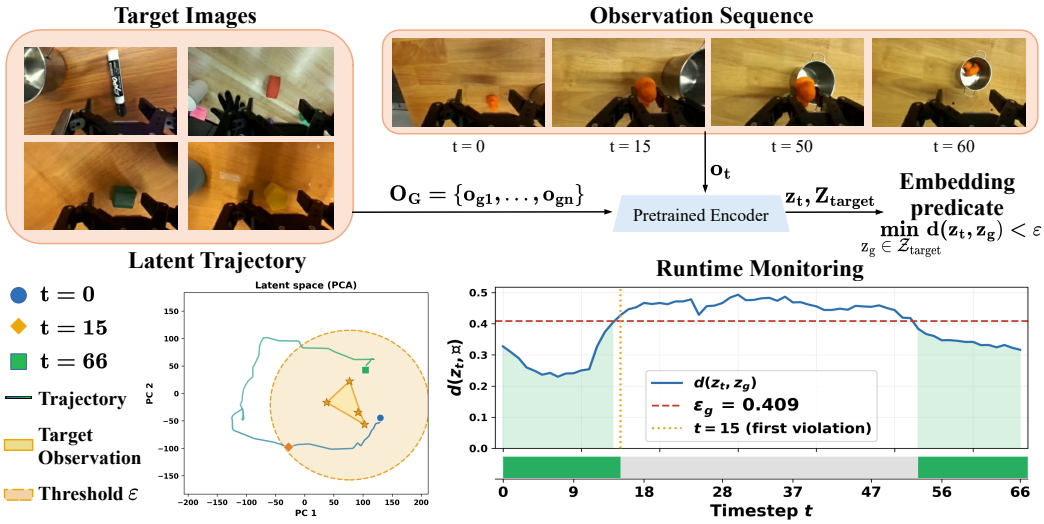


Figure 1: Overview of embedding-based runtime monitoring. **Top:** Target observations $O_G = \{o_{g1}, \ldots, o_{gn}\}$ and the online observation o_t are encoded by a pretrained vision encoder into target embeddings z_G and the current embedding z_t. An embedding predicate then evaluates whether z_t reaches the target set. **Bottom left:** The embedding trajectory is projected onto its first two principal components, showing its evolution relative to the target embeddings. **Bottom right:** Runtime monitoring computes the distance from z_t to z_G over time, thresholds it using ϵ, and evaluates the temporal logic specification.

3 Embedding Temporal Logic

3.1 Syntax and Semantics

In our approach, a perception-based system is assumed to make an observation about the real world through a sensor (e.g., camera) at each step in its execution. This observation is then passed through an encoder that translates the observation into an embedding. We formally model such a system as an *Embedding Temporal Structure*.

Definition 1 (Embedding Temporal Structure)**.** *An* Embedding Temporal Structure *is a tuple*

$$\mathcal{M} \equiv (\mathcal{S}, \mathcal{O}, \mathcal{Z}, \phi_{obs}, \psi_{enc}, D_{\mathcal{Z}}, AP_z),$$

where $\mathcal{S}, \mathcal{O}, \mathcal{Z}$ denote the sets of ground-truth states, observations, and embedding spaces, respectively; $\phi_{obs} : \mathcal{S} \to \mathcal{O}$ is the observation function that maps the ground-truth states to the states that can be observed with the given sensor(s); $\psi_{enc} : \mathcal{O} \to \mathcal{Z}$ is the embedding function that converts an observation into an embedding; $D_{\mathcal{Z}}$ is a set of admissible distance/similarity functions $d : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}_{\geq 0}$; and AP_z is the set of embedding predicates (Definition 5).

Conceptually, z is an approximation of a latent variable; i.e., the state of the world that is only indirectly observable by the system. The *organization* of latent representations of observations within the embedding space $\mathcal{Z}$ is induced by the encoder ψ_{enc} and the downstream models that subsequently consume this embedding for further tasks (e.g., object identification and prediction) and shape the embedding organization as part of their training. This organization of embeddings in turn has a pronounced effect on the distances between embeddings.

To reason about a system's temporal behavior, we must consider not only individual states but sequences of states over time. We therefore define an execution of the system and the corresponding embedding-space representation induced by the observation and encoder mappings.

Definition 2 (Execution)**.** *An execution of the system $\mathcal{M}$ is a finite or infinite sequence of states $\varsigma = s_0, s_1, s_2, \ldots$ such that each state s_t belongs to the state space $\mathcal{S}$ for every time index $t \in \mathbb{N}$.*

Each state in an execution can be mapped through ϕ_{obs} and ψ_{enc}, yielding a corresponding representation in the embedding space.

Definition 3 (Representation Map)**.** *The mapping from the ground-truth states to embeddings is the* representation map $\eta \equiv \psi_{enc} \circ \phi_{obs} : S \to \mathcal{Z}$.

Definition 4 (Trace)**.** *Given an execution* $\varsigma = (s_i)_{i \in \mathbb{N}}$*, the associated* trace *is the sequence* $\sigma = (z_i)_{i \in \mathbb{N}}$*, where* $z_i = \eta(s_i)$ *for all* $i \in \mathbb{N}$.

Embedding Predicates Unlike classical temporal logics where atomic propositions are Boolean predicates over S, ETL predicates are defined over the embedding space $\mathcal{Z}$.

Definition 5 (Embedding Predicate)**.** *An* embedding predicate $ap \in AP_z$ *is a tuple* $ap \equiv (\mathcal{Z}_{target}, d, \epsilon, \bowtie, a)$ *where:* $\mathcal{Z}_{target} \subseteq \mathcal{Z}$ *is a set of target embeddings;* $d \in D_{\mathcal{Z}}$*, where* $d : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}_{\geq 0}$*, is a distance function;* $\epsilon \in \mathbb{R}_{\geq 0}$ *is a threshold; and* a *is an aggregation operator (e.g.,* $\min$*,* $\max$*).*

The set of target embeddings identifies sensor inputs that the system should either reach or avoid in order for the system to either be considered safe or to have reached as a goal. Additional details for specifying target embeddings are discussed in Appendix A.1.

We now define how an Embedding Predicate can be evaluated.

Definition 6 (Predicate Satisfaction)**.** *For a given embedding predicate* $ap \in AP_z$ *and embedding* z*, the evaluation of an embedding predicate is defined as:*

$$\delta_{ap}(z) = a(\{d(z, z_g) \mid z_g \in \mathcal{Z}_{target}\})$$

Then, the satisfaction of the predicate is the Boolean value determined by $\delta_{ap}(z) \bowtie \epsilon$.

Example 1. Consider an embedding predicate ap as illustrated in Figure 1, where the target images depict a desired visual concept (e.g., object not grasped) and are encoded into a target set $\mathcal{Z}_{target}$. Given the embedding z_t of the current observation, predicate evaluation measures the distance between z_t and the target set. Let d be the L2 distance, $a = \min$, and $\bowtie = \leq$. Then,

$$\delta_{ap}(z_t) = \min_{z_g \in \mathcal{Z}_{target}} d(z_t, z_g).$$

Using $a = \min$, we ensure that $\delta_{ap}(z_t)$ is the distance from the current observation to the closest target in the embedding space. Then, the predicate is satisfied if $\delta_{ap}(z_t) \leq \epsilon$, meaning that the current observation embedding lies sufficiently close to at least one target embedding, indicating that the desired visual concept is present.

Now that we have defined embedding predicates, we next define the syntax and semantics of ETL to reason over sequences of embeddings over time.

Definition 7 (ETL Syntax)**.** *The* ETL syntax *of formula* φ *is recursively defined as follows:*

$$\varphi ::= ap \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \boldsymbol{U} \varphi_2$$

As in LTL, the until operator **U** can be used to express the *eventually* (**F**) operator and *always* (**G**) operator: $\mathbf{F}\varphi = \texttt{True}\mathbf{U}\varphi$ and $\mathbf{G}\varphi = \neg\mathbf{F}\neg\varphi$.

Definition 8 (ETL Semantics)**.** *For a given trace* σ*, the* ETL semantics *for timestep* i *are defined similarly to those of LTL, but over embedding traces:*

$$\begin{aligned}
&\sigma, i \models ap \iff \delta_{ap}(z_i) \bowtie \epsilon \\
&\sigma, i \models \neg\varphi \iff \sigma, i \not\models \varphi \\
&\sigma, i \models \varphi_1 \wedge \varphi_2 \iff \sigma, i \models \varphi_1 \text{ and } \sigma, i \models \varphi_2 \\
&\sigma, i \models \varphi_1 \boldsymbol{U} \varphi_2 \iff \exists j \geq i \text{ such that } \sigma, j \models \varphi_2 \text{ and } \forall k \in [i, j), \sigma, k \models \varphi_1
\end{aligned}$$

Robustness of ETL Certain types of temporal logics, such as STL [Maler and Nickovic, 2004], allow for a quantitative notion of satisfaction, called *robustness* [Fainekos and Pappas, 2009], that represents the degree to which the system satisfies or violates a specification. To enable ETL to be used for quantitative monitoring, we also introduce a notion of robustness for ETL. For the sake of brevity, the formal semantics for robustness are provided in Appendix B.

Example 2 Reusing the embedding predicate ap as defined in Example 1, consider the ETL specification $\varphi = \mathbf{G}(ap)$, which requires that the visual concept encoded by predicate ap is always observed. Suppose the system generates the embedding trace $\sigma = z_{12}, z_{13}, z_{14}, z_{15}$, with minimum distances to $\mathcal{Z}_{target}$ as $[\delta_{ap}(z_t)]_{t=12}^{15} = [0.327,\ 0.374,\ 0.403,\ 0.427]$.

Using the robustness semantics for predicates, $\rho(ap, \sigma, t, 15) = \epsilon - \delta_{ap}(z_t)$, and setting $\epsilon = \epsilon_g = 0.409$, the predicate robustness at each timestep is: $[0.082,\ 0.035,\ 0.006,\ -0.018]$. The robustness of the temporal specification is then $\rho(\varphi, \sigma, 0, 15) = \min_{t\in[0,3]} \rho(ap, \sigma, t, 15) = -0.018$. Intuitively, although the system remains close to the object not being grasped for most of the window, the specification is violated because at timestep 15 the embedding moves outside the threshold ϵ, corresponding to the frame $t = 15$ in Figure 1 where the object is grasped.

3.2 ETL Specification Monitors

An ETL monitor evaluates the satisfaction of a specification over an embedding trace for a given system. ETL monitors are defined only over the finite trace of a system in order to be evaluated during the execution of the system, and thus are only defined for specifications that are well-defined over finite traces.

Definition 9 (Finite Trace)**.** *For a given trace σ, a* finite trace $\sigma_{\leq t}$ *is a trace consisting of the first $t+1$ states of σ.*

We refer to a finite trace as a trace when clear from context. An ETL monitor is formally defined over a finite trace of embeddings by the following definition.

Definition 10 (ETL Monitor)**.** *For a trace $\sigma_{\leq t}$, system $\mathcal{M}$, and specification φ, an* ETL Monitor *is $M_\varphi(\sigma_{\leq t}) = (r_0^\varphi, \ldots, r_t^\varphi) \in \{-1, +1\}^{t+1}$, where $r_i^\varphi = \mathrm{sgn}(\rho(\varphi, \sigma_{\leq i}, 0, i))$ for each $0 \leq i \leq t$, with $\mathrm{sgn}(x) = +1$ if $x \geq 0$ and -1 otherwise.*

Intuitively, when the monitor is positive for a timestep, the system satisfies the ETL specification at that timestep. However, when the robustness of the system becomes negative at timestep i, the monitor raises an alert to indicate that the observed execution violates φ over the finite prefix $[0, i]$. A three-valued operational reading of this definition (satisfied, violated, or pending) is given in Appendix C. ETL monitors enable runtime monitoring of visually defined temporal model and evaluate behavior produced by the system without access to the internal decision-making process.

Semantic Correctness We denote an ETL monitor to be *semantically correct* iff it is equivalent with evaluation over ground-truth executions. A formal definition of semantic correctness is provided in Appendix D. This notion of semantic correctness provides the basis for evaluating ETL monitors in practice. In our experiments (Section 5), we approximate semantic correctness by comparing the outputs of the ETL monitor against ground-truth monitors derived from state-based specifications.

3.3 Constructing ETL Specifications in Practice

Utilizing the formal semantics presented in Section 3.1 requires several design choices for concrete instantiation: one must determine how target embeddings are specified, how observations are mapped into the representation space, which distance function is used to compare embeddings, and how predicate thresholds are calibrated in order to align with the intended semantic concept. In practice, target embeddings should be selected by the engineer in the same format as the sensor input; e.g., for a camera, a target would be provided as a reference image. The distance function selection depends on the training objective of the encoder and ideally should align with the organization of the embedded space. The reliability of these predicates depends on whether the chosen representation preserves the intended semantic distinctions. We assess this through reference-conditioned separability, measuring whether satisfying observations are closer to the target embeddings than non-satisfying ones using AUROC and AUPRC. Poor separability indicates that the reference observations, encoder, or distance metric may need revision. Further details on predicate construction are provided in Appendix A.

4 Threshold Calibration for ETL Predicates

The preceding section described how ETL predicates are instantiated in practice through target embeddings, encoders, and distance functions. We now turn to threshold calibration. Recall from

Definition 5 that ETL predicates are evaluated by comparing embedding distances against a threshold ϵ. These thresholds play a critical role: they determine when a continuous similarity measure corresponds to predicate satisfaction. As a result, ETL specifications are not only parameterized by the encoder and distance function, but also by calibrated thresholds that define when a predicate holds. We scope our work to two approaches to calibrate thresholds, F1-optimal and conformal prediction thresholds. The choice of calibration method and its configuration determine the monitor's operating point at deployment, and we discuss them in Appendix E. We now present the threshold calibration methods; for formal definitions of the thresholds, we refer readers to Appendix F.

4.1 F1-Optimal Threshold

We calibrate each predicate threshold on a held-out set, $\mathcal{D}_{cal}$, of trajectories with ground-truth labels per timestep. For each timestep t, we compute the embedding distance d_t to the target embedding and, for a candidate threshold ϵ_1, predict that the predicate holds whenever $d_t \leq \epsilon_1$. We then search over possible threshold values based on the observed calibration distances and select the threshold, ϵ_{F1}, that maximizes the F1 score with respect to the ground-truth labels. Intuitively, this chooses the threshold that best matches the intended concept by balancing false positives and false negatives on the calibration data.

4.2 Conformal Threshold with Recall Guarantee

In safety-critical monitoring, missing a true positive event is often more costly than raising a false alarm. To bias threshold selection toward high recall, we compute an alternative threshold using *split conformal prediction*, ϵ_{CP} [Lei et al., 2017, Angelopoulos and Bates, 2023]. We employ conformal prediction as it allows us to make no assumptions about the underlying distribution of embedding distances and provides a guarantee that transfers directly to deployment-time predicate evaluation.

In this work, we propose conformal *ETL predicate calibration* by adapting conformal prediction to select predicate thresholds from embedding distance scores. Let $\varsigma^1, \ldots, \varsigma^{n_{cal}}$ be calibration demonstrations, disjoint from training, with z_g denoting the target latent embedding with a semantically corresponding ground-truth specification ω. For each calibration, ς^i, a score is computed based on the maximum distance from the frame to the target such that the target satisfies ω. Then, for a user-given error level $\alpha \in (0, 1)$, we sort the calibration scores, compute $k = \lceil (1-\alpha)(n_{cal}+1) \rceil$ and select ϵ_{CP} to be the k-th smallest score. The formal definition and proof for conformal recall guarantees based on calibration demonstrations can be found in Appendix F.

5 Evaluation

We evaluate the semantic correctness of ETL monitors by comparing embedding predicate outputs against ground-truth propositions.[3] All experiments were run on a compute cluster using two NVIDIA GeForce RTX 5090 GPUs, each with 34.2 GB of memory. We distinguish frame-level predicate metrics (precision, recall, F1) from episode-level satisfaction and subgoal-ordering agreement. Unless otherwise stated, results use a fixed calibration/test split. We additionally report results over 20 random trajectory-level resamplings of successful episodes as mean $\pm$ 95% confidence intervals. Appendix E details the split protocols, calibration sizes, class distributions, threshold methods and α values.

5.1 Simple Navigation Dubins Car

First, we evaluate ETL specification based runtime monitoring in a two dimensional navigation task where privileged information about the state space is available, similar to the one proposed in Agrawal et al. [2025]. This fully observable setup allows direct comparison between embedding-based predicates and ground-truth specifications.

Setup The navigation task is defined for a robot that respects discrete-time Dubins car dynamics in a controlled environment. We generate $N = 100$ trajectories using a feedback controller with obstacle avoidance. For each task, we construct pairs of equivalent specifications in the ground-truth state space, ω_i, and in the embedding space, φ_i. We utilize the encoder from the world model used

[3] Artifacts for reproducibility are provided at `https://github.com/cmu-soda/ETLMonitoring`

in Agrawal et al. [2025], which is based on Dreamer [Hafner et al., 2025]. The encoder produces a task-relevant representation space in which distances reflect semantic similarity between observations. We evaluate four specification patterns: *Reach* ($\mathbf{F}A$), *Avoid* ($\mathbf{G}\neg C$), *Reach-Avoid* ($\mathbf{F}A \wedge \mathbf{G}\neg C$), and *Sequential* ($\mathbf{F}(A \wedge \mathbf{F}B)$). A, B, and C denote reaching a goal in the top-right, the top-left, and the bottom-right corners of the environment respectively. For visualization, see the goals in Figure 2a. ETL monitoring is evaluated by comparing the satisfaction of each embedding-based specification φ with the satisfaction of corresponding ground-truth specifications ω on the respective latent and state traces for each timestep of each trace. We report precision, recall, and F1 to assess how closely ETL predicates match ground-truth events. For temporal specifications, we measure trajectory-level satisfaction agreement with the ground-truth monitor; for sequential specifications, we also report ordering accuracy of detected subgoals. Additional details can be found in Appendix G.1.

Results: Can ETL monitors achieve semantic correctness as defined in Definition 15 with respect to ground-truth specifications in a controlled environment? ETL shows strong semantic correctness for atomic predicates, visualized in Figure 2a. Across reach, avoid, and reach-avoid specifications, the F1-optimal monitor achieves F1 scores of 0.80–0.85 with agreement above 96%, indicating that embedding predicates closely match the corresponding state-based events. This implies that embedding-space predicates can serve as faithful proxies for state-based propositions. Additionally, for the sequential specification $A \rightarrow B$, the monitor achieves 100% precision, recall, and ordering agreement at the episode level. Thus, ETL is not limited to detecting isolated semantic events; *ETL can also correctly track their ordering over time.* The difference between per-frame and episode-level scores arises from the evaluation granularity: the Reach-A F1 score of 0.85 reflects frame-level errors near goal boundaries, whereas sequential correctness is evaluated at the episode level, where localized errors do not affect whether A is detected before B across the test trajectories.

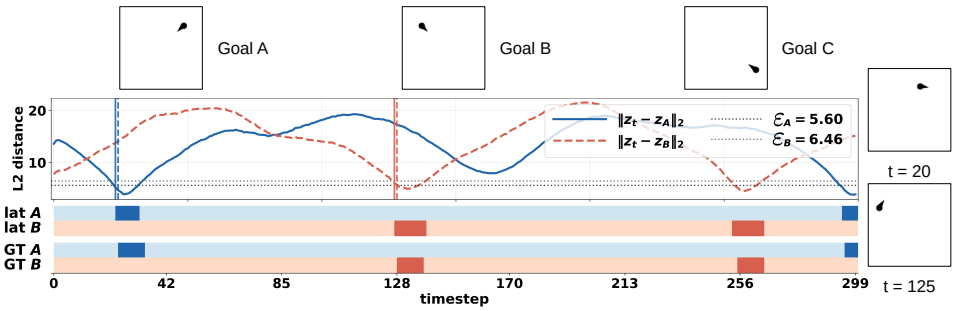
Does threshold calibration produce the intended precision–recall tradeoff? Relative to the F1-optimal threshold, ϵ_{CP} ($\alpha = 0.1$) increases recall for reach (from 0.83 to 0.93) while incurring a drop in precision (from 0.87 to 0.79), with essentially no change in agreement. Conformal calibration thus provides a recall-oriented operating point at the cost of additional false alarms, as expected from the one-sided guarantee.

Overall, the Dubins results show that ETL monitors align closely with state-based specification monitors, support monitoring of temporally composed behaviors, and exhibit a clear precision–recall tradeoff under threshold calibration.

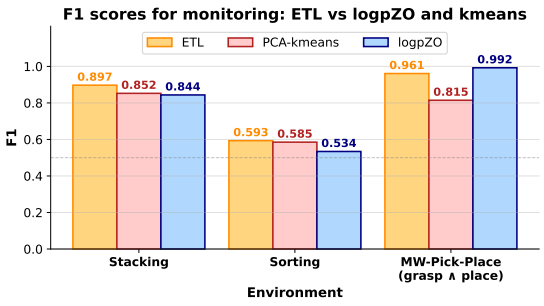
5.2 Simulated Manipulation Tasks

To assess whether ETL remains effective beyond the controlled navigation domain, we evaluate it for simulated contact-rich manipulation tasks with richer visual scenes, complex object interactions, and additional distractor objects that make perceptual monitoring harder. We consider two environments with complementary task structure: D3IL [Jia et al., 2024] and MetaWorld [Nicklas Hansen, 2026]. For D3IL we consider two complex tasks: SORTING requires a Franka robot to push two blocks into their corresponding color-matching target boxes, while STACKING requires the robot to arrange colored blocks in a target region. For MetaWorld, we evaluate ETL on robotic-arm pick-and-place tasks with sequential grasp and place subgoals. Unlike single-goal manipulation tasks, these benchmarks expose phase-like progress structures, making them a natural testbed for evaluating whether ETL predicates can monitor task-relevant semantic milestones in rich manipulation settings.

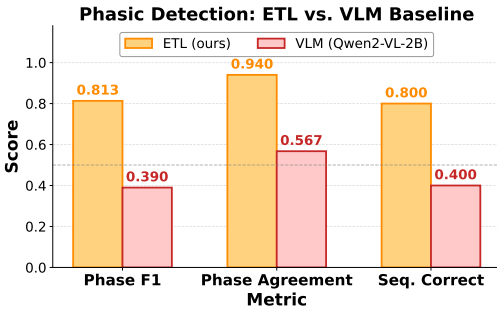
Setup We compare our ETL-based monitoring approach against recent embedding-based monitoring baselines: PCA-kmeans [Liu et al., 2024] and logpZO [Xu et al., 2025]. PCA-kmeans clusters principal components of successful observation embeddings and scores a new observation by its distance to the nearest cluster center, while logpZO fits a flow-matching density model over observation embeddings and uses the inferred latent norm as an uncertainty score. These baselines are the most direct comparison for ETL because they operate on observation embeddings and detect distributional deviations. The embedding-based baselines use only success-labeled calibration embeddings, without target observations or frame-level labels. Both are refitted on each calibration split and evaluated using the same F1-optimal thresholding protocol as ETL. We additionally consider a target-conditioned atomic baseline that shares ETL's encoder, reference observations, labels, and calibration procedure but excludes temporal composition, isolating the contribution of temporal reasoning from calibrated



(a) Boolean predicate traces on the Dubins car. Goals A, B, and C are visualized along top of the image; the graph plots the L2 distance between embeddings and along with the F1-optimal thresholds. The bars along the bottom show when the latent and ground-truth predicates are satisfied (darker bars); two observation states are displayed on the left of the graph.



(b) Predicate-monitoring F1 scores across domains.



(c) Comparison of ETL and Qwen2-VL-2B on phasic DROID episodes.

Figure 2: **Qualitative and quantitative ETL results.** (a) Dubins car trajectories showing embedding distances to goals A, B, and C, F1-optimal thresholds (dashed lines), and ETL versus ground-truth predicate satisfaction (shaded bars). (b) Frame-level F1 comparison of ETL, logpZO, and PCA-kmeans across simulated manipulation tasks. (c) Comparison of ETL and Qwen2-VL-2B on DROID, reporting mean frame-level F1, agreement, and sequential correctness.

embedding similarity. We evaluate monitoring performance by comparing predicted satisfaction or violation labels against ground-truth labels derived from simulator rewards and state variables. Additional details regarding our experimental setup are provided in Appendix H.

Results: Can ETL outperform embedding-based monitoring baselines on simulated manipulation tasks? Figure 2b shows that ETL matches or exceeds the target-agnostic observation-embedding baselines on average across the three simulated manipulation environments. Using ϵ_{F1}, ETL achieves the highest average F1 score compared to logpZO and PCA-kmeans; similarly, ETL improves over the strongest baseline on STACKING and slightly outperforms PCA-kmeans on SORTING. SORTING is the hardest setting for all methods, suggesting that its OOD shifts induce more ambiguous changes in the embedding space. On MetaWorld pick-place-wall, logpZO reaches near-saturated performance (0.992 F1), indicating that the predicate boundary aligns closely with the support of successful embeddings. ETL remains competitive at 0.961 F1, while PCA-kmeans lags behind at 0.815. We additionally conduct a repeated-split evaluation. Under the repeated-split protocol, we randomly partition successful trajectories into calibration and test sets 20 times, while keeping failure trajectories fixed in the test set. We report the mean F1 score and 95% confidence interval across these splits. ETL achieves 0.917 ± 0.004 on STACKING and 0.595 ± 0.012 on SORTING, compared to 0.857 ± 0.011 and 0.515 ± 0.002 for logpZO, respectively, preserving the method rankings. To isolate the benefit of temporal reasoning, we compare ETL against a baseline using identical embedding predicates but without temporal operators. While both achieve comparable generic classification performance, ETL substantially improves detection of failures where subgoals are completed in the wrong order, from 57.7% to 88.5% on STACKING and $\leq 5.0\%$ to 82.7% on SORTING (Appendix H).

5.3 Real World Evaluation via DROID

Finally, we evaluate ETL on real-world robot data to assess its effectiveness for monitoring in-the-wild temporally extended manipulation behaviors. To this end, we utilize the manipulation dataset DROID [Khazatsky et al., 2024], which contains large-scale, diverse, and compositional manipulation demonstrations in more realistic settings.

Setup We analyze manipulation episodes with compositional, multi-phase structure. We instantiate ETL specifications over phase-based predicates, where each phase corresponds to a contiguous interaction window in the demonstration. This setup allows us to evaluate how ETL monitoring performs over real-world manipulation episodes in comparison to a Vision Language Model (VLM)-based baseline monitor. For evaluation, we compute F1 score, agreement, and sequential ordering accuracy over phase-based predicates. For our baseline, we use a state-of-the-art VLM: Qwen2-VL [Wang et al., 2024]. For each phase, the VLM is prompted with the ground-truth desired behavior and phase frame to determine if a task was completed; the response is distilled to a Boolean value, which is then used to compute frame-level F1, agreement, and sequential ordering accuracy for comparison with the ETL monitor. Tabularized results can be found in Appendix I.1.

Results: Can ETL accurately monitor perceptual specifications on real-world manipulation data in comparison to a VLM-based baseline for monitoring phasic manipulation behaviors? With the domain-matched SVD-VAE encoder, ETL achieves a mean frame-level F1 score of 0.813 and mean agreement of 0.940 across the five tasks, whereas the Qwen2-VL baseline reaches a mean F1 score of 0.390 and mean agreement of 0.567. Though ETL and Qwen2-VL perform competitively on one of the five tasks, these results demonstrate that ETL generalizes to diverse, unstructured manipulation scenes, and is markedly more reliable overall, especially on phases that are visually similar but semantically distinct. ETL also recovers the sequential phase ordering on 4 of the 5 task specifications (Figure 2c), indicating it can monitor compositional task structure beyond isolated phase detection. The single failure occurs in an episode with low inter-phase separability, suggesting that monitoring performance degrades when distinct phases occupy overlapping regions of the latent space. To assess whether ETL monitoring generalizes beyond domain-matched representations, we evaluate frozen general-purpose encoders (Appendix I.4), holding targets, labels, thresholds, and specifications fixed to isolate the effect of the representation. While DINOv3 achieves higher frame-level F1 than the domain-matched SVD-VAE (0.834 vs. 0.813), it correctly recovers temporal ordering on only $2/5$ tasks, compared to $4/5$ for SVD-VAE. These results demonstrate that strong semantic discrimination alone is insufficient for reliable temporal monitoring.

Overall, these results demonstrate that ETL extends to real-world manipulation tasks, effectively monitoring sequential behaviors and providing more reliable phase detection than the VLM baseline.

6 Conclusion and Limitations

In this work, we build on prior work [Kapoor et al., 2025a] to extend a temporal logic (ETL) defined over the embedding space of pretrained encoders to runtime monitoring of perception-based autonomous systems. Our experiments support three main conclusions. First, embedding predicates can faithfully approximate ground-truth propositions across controlled navigation and manipulation tasks. Second, ETL supports monitoring temporally extended properties: sequential specifications are accurately tracked across all evaluated environments. Third, threshold calibration provides a meaningful operating tradeoff: the F1-optimal threshold yields the strongest overall alignment with ground truth, while the conformal threshold prioritizes high recall with a distribution-free guarantee.

The current approach has two main limitations: latent predicates are not yet fully interpretable in human-understandable terms, and monitoring performance depends on whether task-relevant semantic concepts are well separated in the encoder's representation. These limitations point to promising directions for future work, including more transparent predicate explanations, encoder selection and adaptation for improved semantic separation Zarlenga et al. [2022], temporal abstraction over subtask boundaries Wang et al. [2026], predictive action-level checking for candidate traces, and adaptive online thresholding Areces et al. [2025].

References

Sankalp Agrawal, Junwon Seo, Kensuke Nakamura, Ran Tian, and Andrea Bajcsy. Anysafe: Adapting latent safety filters at runtime via safety constraint parameterization in the latent space, 2025. URL https://arxiv.org/abs/2509.19555.

Derya Aksaray, Austin Jones, Zhaodan Kong, Mac Schwager, and Calin Belta. Q-learning for robust satisfaction of signal temporal logic specifications, 2016.

Jasmine Jerry Aloor, Jay Patrikar, Parv Kapoor, Jean Oh, and Sebastian Scherer. Follow the rules: Online signal temporal logic tree search for guided imitation learning in stochastic domains. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1320–1326, 2023. doi: 10.1109/ICRA48891.2023.10160953.

Rajeev Alur, Osbert Bastani, Kishor Jothimurugan, Mateo Perez, Fabio Somenzi, and Ashutosh Trivedi. Policy synthesis and reinforcement learning for discounted ltl. In *International Conference on Computer Aided Verification*, pages 415–435. Springer, 2023.

Anastasios N. Angelopoulos and Stephen Bates. Conformal prediction: A gentle introduction. *Found. Trends Mach. Learn.*, 16(4):494–591, March 2023. ISSN 1935-8237. doi: 10.1561/2200000101. URL https://doi.org/10.1561/2200000101.

Felipe Areces, Christopher Mohri, Tatsunori Hashimoto, and John Duchi. Online conformal prediction via online optimization. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=KwGc2JUIDK.

Mustafa Baniodeh, Kratarth Goel, Scott Ettinger, Carlos Fuertes, Ari Seff, Tim Shen, Cole Gulino, Chenjie Yang, Ghassen Jerfel, Dokook Choe, Rui Wang, Vinutha Kallem, Sergio Casas, Rami Al-Rfou, Benjamin Sapp, and Dragomir Anguelov. Scaling laws of motion forecasting and planning – a technical report, 2025. URL https://arxiv.org/abs/2506.08228.

Ezio Bartocci, Jyotirmoy Deshmukh, Alexandre Donzé, Georgios Fainekos, Oded Maler, Dejan Ničković, and Sriram Sankaranarayanan. Specification-Based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications. In Ezio Bartocci and Yliès Falcone, editors, *Lectures on Runtime Verification: Introductory and Advanced Topics*, Lecture Notes in Computer Science, pages 135–175. Springer International Publishing, Cham, 2018. ISBN 978-3-319-75632-5. doi: 10.1007/978-3-319-75632-5_5.

Calin Belta and Sadra Sadraddini. Formal methods for control synthesis: An optimization perspective. *Annual Review of Control, Robotics, and Autonomous Systems*, 2(1):115–140, 2019. doi: 10.1146/annurev-control-053018-023717. URL https://doi.org/10.1146/annurev-control-053018-023717.

Christian Colombo and Gordon J Pace. What is runtime verification. In *Runtime Verification: A Hands-On Approach in Java*, pages 9–15. Springer, 2022.

Adel Dokhanchi, Heni Ben Amor, Jyotirmoy V Deshmukh, and Georgios Fainekos. Evaluating perception systems for autonomous vehicles using quality temporal logic. In *Runtime Verification: 18th International Conference, RV 2018, Limassol, Cyprus, November 10–13, 2018, Proceedings 18*, pages 409–416. Springer, 2018.

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

Yue Gu, William Hunt, Blair Archibald, Mengwei Xu, Michele Sevegnani, and Mohammad D Soorati. Successful swarms: operator situational awareness with modelling and verification at runtime. In *2023 32nd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 541–548. IEEE, 2023.

Yanjiang Guo, Lucy Xiaoyang Shi, Jianyu Chen, and Chelsea Finn. Ctrl-world: A controllable generative world model for robot manipulation, 2026. URL https://arxiv.org/abs/2510.10125.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination, 2020. URL https://arxiv.org/abs/1912.01603.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, Apr 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-08744-2. URL https://doi.org/10.1038/s41586-025-08744-2.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control, 2024. URL https://arxiv.org/abs/2310.16828.

Mohammad Hekmatnejad, Bardh Hoxha, Jyotirmoy V. Deshmukh, Yezhou Yang, and Georgios Fainekos. Formalizing and evaluating requirements of perception systems for automated vehicles using spatio-temporal perception logic. *The International Journal of Robotics Research*, 43(2):203–238, 2024. doi: 10.1177/02783649231223546. URL https://doi.org/10.1177/02783649231223546.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^*_{0.6}$: a vla that learns from experience, 2025. URL https://arxiv.org/abs/2511.14759.

Xiaogang Jia, Denis Blessing, Xinkai Jiang, Moritz Reuss, Atalay Donat, Rudolf Lioutikov, and Gerhard Neumann. Towards diverse behaviors: A benchmark for imitation learning with human demonstrations. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=6pPYRXKPpw.

Parv Kapoor, Abigail Hammer, Ashish Kapoor, Karen Leung, and Eunsuk Kang. Pretrained embeddings as a behavior specification mechanism, 2025a. URL https://arxiv.org/abs/2503.02012.

Parv Kapoor, Kazuki Mizuta, Eunsuk Kang, and Karen Leung. STLCG++: A masking approach for differentiable signal temporal logic specification. *IEEE Robotics and Automation Letters*, 10(9):9240–9247, 2025b.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, Vitor Guizilini, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Muhammad Zubair Irshad, Donovon Jackson, Charlotte Le, Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. Droid: A large-scale in-the-wild robot manipulation dataset. 2024.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Hadas Kress-Gazit, Georgios E Fainekos, and George J Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE transactions on robotics*, 25(6):1370–1381, 2009.

Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J. Tibshirani, and Larry Wasserman. Distribution-free predictive inference for regression, 2017. URL https://arxiv.org/abs/1604.04173.

Karen Leung, Nikos Aréchiga, and Marco Pavone. Backpropagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods. *The International Journal of Robotics Research*, page 02783649221082115, 2023.

Zhenyu Lin and John S. Baras. Planning and runtime monitoring of robotic manipulator using metric interval temporal logic. In *2019 IEEE International Systems Conference (SysCon)*, pages 1–8, 2019. doi: 10.1109/SYSCON.2019.8836933.

Lars Lindemann and Dimos V. Dimarogonas. Control barrier functions for signal temporal logic tasks. *IEEE Control Systems Letters*, 3:96–101, 2019. URL https://api.semanticscholar.org/CorpusID:50767137.

Huihan Liu, Yu Zhang, Vaarij Betala, Evan Zhang, James Liu, Crystal Ding, and Yuke Zhu. Multi-task interactive robot fleet learning with visual world models, 2024. URL https://arxiv.org/abs/2410.22689.

Matt Luckcuck, Marie Farrell, Louise A. Dennis, Clare Dixon, and Michael Fisher. Formal specification and verification of autonomous robotic systems: A survey. *ACM Comput. Surv.*, 52(5), September 2019.

Oded Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In *FORMATS/FTRTFT*, 2004. URL https://api.semanticscholar.org/CorpusID:15642684.

Kensuke Nakamura, Lasse Peters, and Andrea Bajcsy. Generalizing safety beyond collision-avoidance via latent-space reachability analysis. In *Robotics: Science and Systems XXI*, RSS2025. Robotics: Science and Systems Foundation, 2025. doi: 10.15607/rss.2025.xxi.113. URL http://dx.doi.org/10.15607/RSS.2025.XXI.113.

Xiaolong Wang Nicklas Hansen, Hao Su. Learning massively multitask world models for continuous control, 2026.

NVIDIA, :, Arslan Ali, Junjie Bai, Maciej Bala, Yogesh Balaji, Aaron Blakeman, Tiffany Cai, Jiaxin Cao, Tianshi Cao, Elizabeth Cha, Yu-Wei Chao, Prithvijit Chattopadhyay, Mike Chen, Yongxin Chen, Yu Chen, Shuai Cheng, Yin Cui, Jenna Diamond, Yifan Ding, Jiaojiao Fan, Linxi Fan, Liang Feng, Francesco Ferroni, Sanja Fidler, Xiao Fu, Ruiyuan Gao, Yunhao Ge, Jinwei Gu, Aryaman Gupta, Siddharth Gururani, Imad El Hanafi, Ali Hassani, Zekun Hao, Jacob Huffman, Joel Jang, Pooya Jannaty, Jan Kautz, Grace Lam, Xuan Li, Zhaoshuo Li, Maosheng Liao, Chen-Hsuan Lin, Tsung-Yi Lin, Yen-Chen Lin, Huan Ling, Ming-Yu Liu, Xian Liu, Yifan Lu, Alice Luo, Qianli Ma, Hanzi Mao, Kaichun Mo, Seungjun Nah, Yashraj Narang, Abhijeet Panaskar, Lindsey Pavao, Trung Pham, Morteza Ramezanali, Fitsum Reda, Scott Reed, Xuanchi Ren, Haonan Shao, Yue Shen, Stella Shi, Shuran Song, Bartosz Stefaniak, Shangkun Sun, Shitao Tang, Sameena Tasmeen, Lyne Tchapmi, Wei-Cheng Tseng, Jibin Varghese, Andrew Z. Wang, Hao Wang, Haoxiang Wang, Heng Wang, Ting-Chun Wang, Fangyin Wei, Jiashu Xu, Dinghao Yang, Xiaodong Yang, Haotian Ye, Seonghyeon Ye, Xiaohui Zeng, Jing Zhang, Qinsheng Zhang, Kaiwen Zheng, Andrew Zhu, and Yuke Zhu. World simulation with video foundation models for physical ai, 2026. URL https://arxiv.org/abs/2511.00062.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024. URL https://arxiv.org/abs/2304.07193.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

Vasumathi Raman, Alexandre Donze, Mehdi Maasoumy, Richard M Murray, Alberto Sangiovanni-Vincentelli, and Sanjit Seshia. Model predictive control with signal temporal logic specifications. In *Conference on Decision and Control*, 2014.

Oliver Schön and Lars Lindemann. Spatiotemporal robustness of temporal logic tasks using multi-objective reasoning. *arXiv preprint arXiv:2603.29868*, 2026.

Sanjit A. Seshia, Ankush Desai, Tommaso Dreossi, Daniel J. Fremont, Shromona Ghosh, Edward Kim, Sumukh Shivakumar, Marcell Vazquez-Chanlatte, and Xiangyu Yue. Formal specification for deep neural networks. In *Automated Technology for Verification and Analysis*, pages 20–34, 2018.

Dawei Sun, Jingkai Chen, Sayan Mitra, and Chuchu Fan. Multi-agent motion planning from signal temporal logic specifications. *IEEE Robotics and Automation Letters*, PP:1–1, 2022. URL https://api.semanticscholar.org/CorpusID:245986629.

Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer-Verlag, Berlin, Heidelberg, 2005. ISBN 0387001522.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution, 2024. URL https://arxiv.org/abs/2409.12191.

Ying Wang, Oumayma Bounou, Gaoyue Zhou, Randall Balestriero, Tim G. J. Rudner, Yann LeCun, and Mengye Ren. Temporal straightening for latent planning, 2026. URL https://arxiv.org/abs/2603.12231.

Chen Xu, Tony Khuong Nguyen, Emma Dixon, Christopher Rodriguez, Patrick Miller, Robert Lee, Paarth Shah, Rares Ambrus, Haruki Nishimura, and Masha Itkina. Can we detect failures without failure data? uncertainty-aware runtime failure detection for imitation learning policies, 2025. URL https://arxiv.org/abs/2503.08558.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies, 2026. URL https://arxiv.org/abs/2602.15922.

Mateo Espinosa Zarlenga, Pietro Barbiero, Gabriele Ciravegna, Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, Zohreh Shams, Frederic Precioso, Stefano Melacci, Adrian Weller, Pietro Lio, and Mateja Jamnik. Concept embedding models: Beyond the accuracy-explainability trade-off, 2022. URL https://arxiv.org/abs/2209.09056.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning, 2025. URL https://arxiv.org/abs/2411.04983.

A Constructing ETL Specifications in Practice

The formal semantics proposed in the preceding section define ETL as a specification language over embedding traces. To use ETL in practice, however, several design choices must be instantiated concretely: one must determine how target embeddings are specified, how observations are mapped into the representation space, which distance function is used to compare embeddings, and how predicate thresholds are calibrated so that they align with the intended semantic concept. In this section, we first outline how target embeddings are specified. Then, we highlight the different choices of encoders and distance functions.

A.1 Specifying Target Embeddings

ETL is specified using predicates over *target embeddings* that correspond to parts of the physical world against which current observations are evaluated. In practice, the specifier (e.g., a system engineer) would not directly specify the mathematical representations of embeddings. Instead, the targets would be provided in the same format as the sensor's input, e.g., for a camera, a target would be provided as a reference image, that is then translated into the target embeddings via the pretrained encoder. This allows specifications such as *"eventually reach a state similar to this image"* or *"always avoid states resembling fire"* without requiring explicit symbolic labels. By enabling specification directly over learned representations rather than explicit symbolic labels, our approach removes the need to manually define a finite predicate vocabulary from observations, a process that has traditionally depended on substantial expert knowledge and task-specific engineering in robotics.

The use of a few reference observations to represent a semantic concept is well established in few-shot classification, nearest-neighbor retrieval with pretrained representations and image-goal-conditioned robotics. In these methods, a pretrained representation provides prior semantic structure and invariances, while the reference observation grounds the desired concept within that representation.

A concept may instead be multimodal or dominated by nuisance variation. Large-scale pretraining partially mitigates these concerns by learning invariances from diverse data and augmentation objectives; pretraining does not guarantee that every semantic predicate is recoverable through distance to a selected anchor. ETL therefore relies on pretrained invariances as a prior; the properties required of the encoder, distance, references, and calibration set, and the diagnostics used to test them, are given in Section A.3.

A.2 Choice of Encoders and Distance functions

The choice of distance function is another key practical consideration in evaluating ETL satisfaction. Distances over pretrained image embeddings capture perceptual similarity, but lack temporal context because these encoders are trained on static observations. World models mitigate this limitation by mapping observations into a latent space that evolves as a function of past states and actions. When trained with reconstruction-based objectives, such models often preserve much of the geometry of the original embedding space while enriching it with temporal structure. Consequently, similarity relationships defined over observation embeddings can often be carried over to the latent space in a meaningful way.

Accordingly, the choice of d should align with the geometry induced by the representation. For instance, cosine distance is a natural choice for contrastive embeddings, whereas L2 distance is often more appropriate for reconstruction-based latent representations. More generally, the selected distance should be one that best reflects semantic similarity in the underlying space. We empirically study the effect of this design choice in Section 5.

A.3 Predicate Construction Procedure

We construct each embedding predicate through the following procedure:

1. *Reference selection.* The engineer provides a set of labeled reference observations representing the desired semantic concept (e.g., images depicting a successful grasp). These observations are encoded using ψ_{enc} to construct the target embedding set $\mathcal{Z}_{target}$.
2. *Calibration data.* A separate set of demonstrations, disjoint from the reference observations and encoder training data, is labeled at each timestep according to the ground-truth satisfaction of the predicate. Labels may be obtained from simulator states, proprioceptive measurements, or manual annotations (Appendix I.2).
3. *Distance computation.* Each calibration observation is encoded, and its distance to $\mathcal{Z}_{target}$ is computed using the selected distance metric d and aggregation operator a. These choices should reflect the structure of the representation space, such as cosine distance for contrastive embeddings or Euclidean distance for reconstruction-based latents.
4. *Threshold calibration.* The predicate threshold ϵ is estimated from the labeled calibration data using either the F1-optimal or split-conformal procedure described in Section 4.
5. *Held-out validation.* The calibrated predicate is evaluated on held-out demonstrations to assess its ability to distinguish satisfying from non-satisfying observations. Predicates with inadequate discrimination are rejected, and their reference observations, encoder, or distance metric are revised.

The reliability of this procedure depends on the suitability of the underlying representation. In particular, the encoder should preserve task-relevant semantic distinctions while remaining robust to nuisance variations such as viewpoint, lighting, and background. The distance metric and aggregation operator should assign smaller distances to satisfying observations than to non-satisfying ones, while the reference set should adequately represent the visual diversity of the target concept. Furthermore, calibration demonstrations should be representative of the intended deployment distribution. Because these assumptions cannot be verified from the encoder alone, we test them with two *viability diagnostics*: (i) *reference-conditioned separability*, where the same encoder, references, distance, and aggregation used by the monitor produce target-distance scores on held-out positive and negative frames, and we report AUROC and AUPRC; low values indicate that the encoded observations do not define a reliable neighborhood for the concept and the predicate is rejected; and (ii) *threshold viability*, where for both F1-optimal and conformal calibration we report precision, recall, false-positive rate, and threshold acceptance rate on held-out data, so that an operating point can be chosen according to the tolerance for misses and false alarms.

A.4 Sensitivity to the Number of Reference Observations

The target set $\mathcal{Z}_{target}$ is built from a small number of reference episodes. Table 1 reports DROID grasp-detection accuracy (conformal threshold, $\alpha = 0.2$) as the number of reference episodes used to construct the target set grows. Accuracy improves monotonically and saturates by 8–12 references, indicating that a handful of references suffices when they cover the visual modes of the concept.

Table 1: DROID grasp detection accuracy versus number of reference episodes (conformal threshold, $\alpha = 0.2$).

Reference episodes	2	4	8	12
Accuracy (%)	83.2	84.8	87.6	88.7

B Quantitative semantics of ETL

Definition 11 (ETL Robustness)**.** *For a trace σ, the* robustness of an ETL formula *at timestep i is defined as follows:*

$$
\begin{aligned}
\rho(ap, \sigma, i, \boldsymbol{b}) &= \begin{cases} \epsilon - \delta_{ap}(z_i) & \bowtie \in \{\leq, <\} \\ \delta_{ap}(z_i) - \epsilon & \bowtie \in \{\geq, >\} \end{cases} \\
\rho(\neg\varphi, \sigma, i, \boldsymbol{b}) &= -\rho(\varphi, \sigma, i, \boldsymbol{b}) \\
\rho(\varphi_1 \wedge \varphi_2, \sigma, i, \boldsymbol{b}) &= \min\big(\rho(\varphi_1, \sigma, i, \boldsymbol{b}), \rho(\varphi_2, \sigma, i, \boldsymbol{b})\big) \\
\rho(\boldsymbol{G}\varphi, \sigma, i, \boldsymbol{b}) &= \boldsymbol{inf}_{k \in [i, \boldsymbol{b}]} \rho(\varphi, \sigma, k, \boldsymbol{b}) \\
\rho(\boldsymbol{F}\varphi, \sigma, i, \boldsymbol{b}) &= \boldsymbol{sup}_{k \in [i, \boldsymbol{b}]} \rho(\varphi, \sigma, k, \boldsymbol{b})
\end{aligned}
$$

*where **inf** and **sup** are the infimum and supremum operators, respectively. Note that the computation of the satisfaction score is restricted to a subsequence of σ between i to $\boldsymbol{b}$ (i.e., $z_i, z_{i+1}, ..., z_{\boldsymbol{b}}$). The bound $\boldsymbol{b}$ may be determined by, for example, the planning horizon used by a planner (i.e., the length of action sequence used*

*by the planner for behavioral prediction). For brevity, the definition for the until (**U**) operator is omitted, but similar to the one described in Fainekos and Pappas [2009].*

C Finite-Prefix Verdicts

Definition 10 provides a binary evaluation of the observed execution prefix, treating it as a complete finite trace. For online monitoring, we additionally distinguish three operational verdicts: *satisfied*, *violated*, and *pending*. Given a terminal horizon, a specification is satisfied (respectively, violated) when all possible continuations of the observed prefix satisfy (respectively, violate) it; otherwise, its verdict remains pending. For example, an eventuality remains pending until its target is observed or the horizon expires, whereas an always property becomes violated immediately upon observing a counterexample. At the terminal horizon, these verdicts coincide with the binary finite-trace semantics.

D Semantic Correctness of ETL Monitors

To compare ETL monitors to ground-truth executions, we define a ground-truth monitor over finite executions.

Definition 12 (Finite Execution)**.** *For a given execution ς, a* finite execution $\varsigma_{\leq t}$ *is an execution consisting of the first $t+1$ states of ς.*

We refer to a finite execution as an execution when clear from context.

Definition 13 (Ground-Truth Monitor)**.** *Let ω be a temporal logic formula over* symbolic *predicates, and let $\varsigma_{\leq t}$ be a finite execution prefix. The* ground-truth monitor *for ω on $\varsigma_{\leq t}$ is the binary-valued trace*

$$GT_\omega(\varsigma_{\leq t}) = \left(b_0^\omega, b_1^\omega, \ldots, b_t^\omega\right) \in \{-1, +1\}^{t+1},$$

where, for each prefix $\varsigma_{\leq i}$ with $0 \leq i \leq t$,

$$b_i^\omega = \begin{cases} +1 & \text{if } \varsigma_{\leq i} \models \omega, \\ -1 & \text{otherwise.} \end{cases}$$

Here, satisfaction $\varsigma_{\leq i} \models \omega$ is defined by the semantics of the underlying temporal logic over symbolic predicates.

We define the semantic correctness of monitor M using the following definitions.

Definition 14 (Semantic Correspondence)**.** *Let M be an embedding temporal structure with representation map $\eta = \psi_{\text{enc}} \circ \phi_{\text{obs}}$. Let ω be a specification over ground-truth state trajectories $\varsigma_{\leq t}$ and let φ be an ETL specification over finite embedding traces $\sigma_{\leq t}$. We say that φ* corresponds semantically *to ω with respect to M if, for every finite execution prefix $\varsigma_{\leq t} = (s_0, \ldots, s_t)$, letting $\sigma_{\leq t} = (\eta(s_0), \ldots, \eta(s_t))$, we have $\sigma_{\leq t} \models \varphi \iff \varsigma_{\leq t} \models \omega$.*

Intuitively, we say that φ is a latent space specification *semantically corresponds* with the ground-truth specification ω if both are intended to capture the same behavioral property, with ω defined over state-based predicates and φ defined over embedding-space predicates.

Definition 15 (Semantic Correctness of an ETL Monitor)**.** *Let ω be a ground-truth specification and let φ be an ETL specification that corresponds semantically to ω. An ETL monitor M is* semantically correct *with respect to (φ, ω) if, for every execution prefix, $M_\varphi(\sigma_{\leq t}) = GT_\omega(\varsigma_{\leq t})$.*

E Threshold Calibration Target-Set Construction

Target sets and predicates are constructed and validated as described in Appendix A.3; labels are obtained from available signals, such as proprioceptive signals, or from manual annotations (Appendix I.2). Here we document the calibration protocol used for each experiment, the choice of α, and the deployment consequences of the threshold rule.

E.1 Calibration Protocol per Configuration

Table 2 lists, for each experimental configuration, the split rule, the number of calibration demonstrations n_{cal}, the class composition of the test set, the threshold method, the conformal level α, and the number of repetitions. The deterministic protocol uses a single fixed calibration/test split; the repeated-split protocol resamples the successful episodes into calibration and test sets at the trajectory level 20 times, keeping all failure episodes in the test set, and reports mean $\pm$ 95% confidence interval. Under the repeated-split protocol the calibration sets for SORTING and STACKING grow from 50 to 122 and 146 successful episodes, which removes those episodes

from the test set and changes failure prevalence from $145/400$ to $145/328$ and from $485/800$ to $485/704$, respectively.

Table 2: Calibration and evaluation protocol per configuration.

Configuration	Split rule	n_{cal}	Test composition	Threshold	α	Reps.
Dubins (all specs)	fixed 40/60 traj. split	40	60 traj., per-frame labels	F1-opt., CP	0.10	1
MetaWorld grasp/place	fixed traj. split	20	24 traj., per-frame labels	F1-opt., CP	0.10	1
SORTING (deterministic)	fixed, success-only cal.	50	145 fail / 255 succ.	F1-opt., CP	0.10	1
STACKING (deterministic)	fixed, success-only cal.	50	485 fail / 315 succ.	F1-opt., CP	0.10	1
SORTING (repeated)	random traj.-level, success-only cal.	122	145 fail / 183 succ.	F1-opt.	–	20
STACKING (repeated)	random traj.-level, success-only cal.	146	485 fail / 219 succ.	F1-opt.	–	20
DROID sequential (hold→release)	fixed traj. split	10	15 traj.	F1-opt., CP	0.20	1
DROID phasic (5 tasks)	fixed traj. split per task	10	held-out traj. per task	F1-opt.	–	1
DROID coverage sweep	random traj.-level	10	held-out traj.	CP	0.1–0.2	20

E.2 Hold-Out Set Selection Procedure

The selection procedure for separating reference and calibration images follows standard procedures for learned latent safety filtering Nakamura et al. [2025], where analytically specifying constraints such as spilling, toppling, or other visually identifiable failures is challenging or impossible in some cases. Prior latent safety filters represent failure sets using classifiers trained from safe/failure observations, and recent image-conditioned filters such as AnySafe Agrawal et al. [2025] specify a runtime constraint using a reference failure image and define the effective latent failure set through a calibrated similarity threshold. The goal in these settings is not exact recovery of a privileged state-space boundary (which is generally unavailable) but an empirical latent approximation.

E.3 Ablations on Alpha for Conformal Thresholds

We conduct additional experiments varying α for conformal threshold calibration and report empirical per-demonstration coverage (the fraction of held-out demonstrations in which every ground-truth-positive timestep is detected) against the corresponding nominal level $1 - \alpha$. Table 3 shows that at $\alpha = 0.2$ (nominal 0.8) every configuration meets or exceeds the nominal level, including the frozen general-purpose encoders on DROID, and that at $\alpha = 0.1$ (nominal 0.9) the MetaWorld predicates also exceed it.

Choosing α and the threshold rule The level α is the tolerated probability of imperfect recall on a new demonstration, and the smallest non-vacuous level is limited by n_{cal}: with n_{cal} calibration demonstrations, any $\alpha < 1/(n_{cal} + 1)$ returns the sample maximum, which is fragile. We therefore keep each benchmark's original operating point ($\alpha = 0.1$ for Dubins and MetaWorld, Table 2) and use $\alpha = 0.2$ for the DROID sequential tasks, where $n_{cal} = 10$ makes $\alpha = 0.1$ coincide with the sample maximum.

Configuration	$\alpha = 0.1$ (nominal 0.9)	$\alpha = 0.2$ (nominal 0.8)
DROID, DINOv2		0.827 ± 0.034
DROID, CLIP		0.824 ± 0.033
DROID, SVD-VAE		0.901 ± 0.026
MetaWorld grasp	0.928	0.818 ± 0.034
MetaWorld place	0.985	0.803 ± 0.034

Table 3: Empirical per-demonstration coverage of the conformal threshold for varying α (mean $\pm$ 95% CI over repeated splits where available).

E.4 Effects of Choice of Threshold on Deployment

Selection of threshold calibration methods is part of the configuration process determined by the developer; we present guidelines below to inform a decision between conformal thresholding and F1-optimal thresholding.

- Use the split-conformal threshold when missed detections are the more costly error and a sufficiently large labeled positive calibration set is available.
- Use the F1-optimal threshold when false positives and false negatives should be balanced according to empirical classification performance.

To evaluate how this choice affects deployment, we simulated an online halting monitor for the SORTING task. The results in Table 4 show that the threshold rule changes the operational tradeoff between needless halts on correct runs and wrong-order failures caught: tightening the conformal level from $\alpha = 0.1$ to $\alpha = 0.05$ raises the fraction of failures caught from 0.904 to 0.971 but also raises needless halts from 0.169 to 0.288, while oracle predicates would achieve 0 and 1.

Threshold rule	Needless halts on correct runs	Wrong-order failures caught
F1-optimal	0.186	0.913
Conformal, $\alpha = 0.1$	0.169	0.904
Conformal, $\alpha = 0.05$	0.288	0.971
Oracle predicates	0.000	1.000

Table 4: Online halting monitor for the SORTING task. The monitor checks the property "blue must not be sorted before red" and halts when the blue predicate fires before the red predicate has occurred. All methods use the same encoder, targets, distances, and temporal specification; only the threshold rule changes.

F Formal Definitions of Threshold Calibration

We provide additional details and intuitive understanding of thresholds and their calibration for ETL monitoring.

Intuitively, ϵ defines the *boundary of a concept* in the representation space (e.g., how close an observation must be to a goal embedding to count as "reached"). Unlike state-based predicates, thresholds in embedding spaces depend on the geometry induced by the encoder and the choice of distance function. Poorly chosen thresholds can lead to high false negatives (overly strict predicates) or false positives (overly permissive predicates), making calibration essential.

We propose a data-driven approach to calibrate thresholds for ETL predicates. We assume access to a dataset of trajectories (e.g., simulator rollouts or demonstrations) with ground-truth signals $\varsigma_{\leq t}$ that satisfy a TL specification ω at timestep t in the underlying state space. Here, ω is a TL specification that semantically corresponds to the ETL specification being instantiated. Then, each trajectory is mapped to an embedding trace using the representation map η, yielding embeddings $z_t = \eta(s_t)$. From these embeddings, we compute distances to the target set as $d_t = d_u(z_t, T_u)$. We then select thresholds using a held-out calibration set of size n_{cal}.

F.1 F1-Optimal Threshold

Definition 16 (F1-Optimal Threshold)**.** *Let $\mathcal{D}_{\text{cal}} = \{(d_t, y_t)\}_{t=1}^{N}$ be the calibration set, where $d_t \in \mathbb{R}_{\geq 0}$ is the embedding distance at timestep t and $y_t \in \{0, 1\}$ is the corresponding ground-truth predicate label. For any candidate threshold $\epsilon \in \mathbb{R}_{\geq 0}$, define the induced prediction*

$$\hat{y}_t(\epsilon) = \mathbf{1}[d_t \leq \epsilon].$$

Let

$$\mathrm{F1}(\epsilon) = \frac{2\,\mathrm{TP}(\epsilon)}{2\,\mathrm{TP}(\epsilon) + \mathrm{FP}(\epsilon) + \mathrm{FN}(\epsilon)},$$

where $\mathrm{TP}(\epsilon)$, $\mathrm{FP}(\epsilon)$, and $\mathrm{FN}(\epsilon)$ denote the numbers of true positives, false positives, and false negatives, respectively, obtained by comparing $\hat{y}_t(\epsilon)$ against y_t over $\mathcal{D}_{\text{cal}}$. The F1-optimal threshold *is defined as*

$$\epsilon_{F1} \in \arg\max_{\epsilon \in \{d_1, \ldots, d_N\}} \mathrm{F1}(\epsilon).$$

If multiple thresholds achieve the same maximum F1 score, ties may be broken arbitrarily.

F.2 Conformal Threshold with Recall Guarantee

The goal of conformal ETL predicate calibration is not to produce a prediction interval around a model output, but to calibrate the threshold used for ETL predicate evaluation. We do this in two stages. Within each calibration trajectory, we compute a single score from all satisfying timesteps by taking the largest embedding distance among the frames that satisfy the predicate. This score represents the *hardest positive case* in that trajectory: any threshold smaller than this value would miss at least one true positive frame in that same trajectory. Then across trajectories, we collect one such score from each calibration trajectory and apply split conformal prediction to these held-out scores. The resulting threshold ϵ_{CP} is therefore calibrated not to individual frames, but between different trajectories. As a result, for a new trajectory drawn from the same distribution, with probability at least $1 - \alpha$, the threshold is large enough to cover that trajectory's hardest positive case, and hence will detect every ground-truth positive timestep in that trajectory. Here, $\alpha \in (0, 1)$ is a user-specified error level.

Setup Let $\varsigma^1, \ldots, \varsigma^{n_{\text{cal}}}$ be n_{cal} calibration demonstrations, disjoint from training, and let z_g denote the target latent embedding that semantically corresponds to a ground-truth specification ω (see Definition 14). For each calibration demonstration ς^i, define the nonconformity score

$$score_i = \max_{t \in [0, |\varsigma^i|]} \begin{cases} d(z_t^{(i)}, z_g) & \text{if } \delta_\omega^{GT}(s_t^{(i)}) = 1 \\ -\infty & \text{otherwise.} \end{cases}$$

for a distance function $d \in D_{\mathcal{Z}}$ and where $\delta_\omega^{GT}(s_t^{(i)}) = 1$ indicates that the ground-truth state at time t satisfies predicate ω. Intuitively, $score_i$ is the most difficult positive timestep in demonstration ς_i: any threshold below $score_i$ would fail to classify at least one ground-truth positive frame in that demonstration. Each calibration demonstration contains at least one ground-truth positive timestep.

Definition 17 (Conformal Calibration Threshold)**.** *For a calibration set of size n_{cal} and a target mis-recall level $\alpha \in (0, 1)$, a* conformal calibration threshold *ϵ_{CP} is a value in $\mathbb{R}_{\geq 0}$ defined as follows:*

$$\epsilon_{\text{CP}} = score_{(k)}, \qquad k = \lceil (1 - \alpha)(n_{\text{cal}} + 1) \rceil,$$

where $score_{(k)}$ denotes the k-th smallest value among $\{score_i\}_{i=1}^{n_{\text{cal}}}$.

Theorem 1 (Conformal Recall Guarantee)**.** *Assume the calibration demonstrations and a future test demonstration are exchangeable. Then*

$$\mathbb{P}\Big(\forall t : \ \delta_\omega^{GT}(s_t^{(i)}) = 1 \ \Rightarrow \ d(z_t, z_\omega) \leq \epsilon_{\text{CP}}\Big) \geq 1 - \alpha.$$

Equivalently, with probability at least $1 - \alpha$ over a newly drawn test demonstration, the embedding predicate achieves perfect per-demonstration recall, i.e., it detects every labeled-positive timestep of that demonstration.

What the guarantee states Theorem 1 is a *one-sided, predicate-level, per-demonstration recall* guarantee: it bounds the probability that an entire demonstration contains at least one missed positive.

Proof sketch. By exchangeability, the augmented set of scores

$$score_1, \ldots, score_{n_{\text{cal}}}, score_{\text{test}}$$

is exchangeable, where

$$score_{\text{test}} = \max_{\substack{t \in \varsigma_{\text{test}} \\ \delta_\omega^{GT}(s_t) = 1}} d(z_t, z_\omega).$$

The standard split conformal order-statistic argument implies

$$\mathbb{P}\big(score_{\text{test}} \leq score_{(k)}\big) \geq \frac{k}{n_{\text{cal}} + 1} \geq 1 - \alpha.$$

By construction, the event $score_{\text{test}} \leq \epsilon_{\text{CP}}$ is exactly the event that every positive timestep in the test demonstration satisfies $d(z_t, z_\omega) \leq \epsilon_{\text{CP}}$, which proves the recall guarantee. □

At deployment, the predicate is evaluated solely by checking whether

$$d(z_t, z_\omega) \leq \epsilon_{\text{CP}},$$

The guarantee holds for any embedding function and any underlying data distribution, provided the calibration and test demonstrations remain exchangeable.

G Additional Evaluation Details for Dubins Car

G.1 Details for Dubins Car Setup

The navigation task is defined for a robot that respects discrete-time Dubins car dynamics. A state is defined as $s = [p^x, p^y, \theta]$ where $s_{t+1} = s_t + \Delta t[v\cos(\theta_t), v\sin(\theta_t), a_t]$ with continuous angular velocity $a_t \in A = [-a_{\max}, a_{\max}]$. We fix $a_{\max} = 1.25$ rad/s and $v = 1$ m/s, with discretization at $\Delta t = 0.05s$. We generate $N = 100$ trajectories using a feedback controller with obstacle avoidance. For each task, we construct pairs of equivalent specifications in the ground-truth state space and the embedding space. The ground-truth specifications, ω_i, are defined over the state space of the car (e.g., the position p^x, p^y, angular velocity y, etc.). In parallel, the ETL specifications, φ_i, are defined over the embedded images observed during simulation; target embeddings are obtained from the simulated observations corresponding to selected goal states. We utilize the encoder from the world model used in Agrawal et al. [2025], which is based on Dreamer [Hafner et al., 2025] with a Recurrent State Space Model (RSSM). The encoder produces a task-relevant representation space in which distances reflect semantic similarity between observations.

Table 5: ETL predicate evaluation on the Dubins car. Atomic predicate scores are computed at the frame level, where each timestep is classified independently; sequential satisfaction is computed as the episode level, where satisfaction is dependent on ordering detected within the trajectory.

		ϵ^* (F1-optimal)				ϵ_{CP} (α=0.10)		
Spec	Scope	F1	Prec.	Rec.	Agree	Prec.	Rec.	Agree
Reach A	frames	0.85	0.87	0.83	98.6%	0.79	0.93	98.5%
Avoid C	frames	0.80	0.69	0.94	96.5%	0.70	0.92	96.5%
RA $A \wedge \neg C$	frames	0.85	0.87	0.83	98.6%	0.79	0.93	98.5%
Seq. $A \to B$	episodes	1.00	1.00	1.00	100%	1.00	1.00	100%

Table 6: Repeated-split protocol: F1 (mean $\pm$ 95% CI over 20 trajectory-level resamplings of successful episodes; failures fixed in the test set) at the F1-optimal threshold. The deterministic value from Table 8 is shown for reference; the two differ because calibration size and failure prevalence differ between protocols (Appendix E). logpZO refits its density model on every split. PCA-kmeans and MetaWorld were not rerun under this protocol.

	STACKING		SORTING	
Method	deterministic	repeated (20)	deterministic	repeated (20)
ETL (F1-opt.)	0.897	0.917 ± 0.004	0.593	0.595 ± 0.012
logpZO	0.844	0.857 ± 0.011	0.534	0.515 ± 0.002
PCA-kmeans	0.852	not rerun	0.585	not rerun

We evaluate four specification patterns: *Reach* ($\mathbf{F}A$), *Avoid* ($\mathbf{G}\neg C$), *Reach-Avoid* ($\mathbf{F}A \wedge \mathbf{G}\neg C$), and *Sequential* ($\mathbf{F}(A \wedge \mathbf{F}B)$). A, B, and C denote reaching the top-right goal $|p_t - (0.8, 0.8)| < 0.25$, reaching the top-left goal $|p_t - (-0.8, 0.8)| < 0.25$, and entering the obstacle proximity zone $|p_t - (0.8, -0.8)| < 0.5$, respectively. This setup enables direct comparison between embedding-based satisfaction and ground-truth logical semantics. Thresholds are calibrated on a 40/60 calibration/test split using both F1-optimal and conformal procedures. In all experiments, we use $\alpha = 0.10$ for computing ϵ_{cp}.

G.2 Results for Dubins Car

The results for the experiments with the Dubins car can be found in Table 5, including F1, Precision, Recall, and Agreement scores for both F1-Optimal and Conformal Prediction thresholds.

H Additional Details for Manipulation Tasks

We provide qualitative visualizations of ETL predicate behavior on manipulation tasks. These plots illustrate how embedding distances evolve over time and how predicate satisfaction aligns with ground-truth signals.

H.1 Tabularized Results

We present tabularized results from the experiments from Section 5.2 in Table 7 and Table 8 (deterministic protocol), and in Table 6 (repeated-split protocol).

H.2 Temporal Composition versus Target-Conditioned Similarity

Table 9 compares ETL against a target-conditioned atomic baseline. For generic episode classification, both achieve comparable F1 scores on STACKING (0.917 vs. 0.918), while ETL slightly improves performance on SORTING (0.595 vs. 0.562). However, temporal composition provides substantial gains in detecting wrong-order failures, where all subgoals are reached but in an incorrect sequence. ETL detects 88.5% and 82.7% of these failures on STACKING and SORTING, respectively, compared to 57.7% and at most 5.0% for the order-blind baseline. These results demonstrate that ETL's primary advantage lies in monitoring temporal relationships rather than individual goal satisfaction.

Table 7: ETL predicate evaluation on MetaWorld (test split). Frame-level metrics are reported for all predicates.

Task	F1	Prec.	Rec.	Agree	CP Rec.	CP Prec.
Grasp (A)	0.985	0.971	0.999	97.3%	0.883	0.976
Place (B)	0.948	0.901	1.000	91.4%	0.916	0.926
Sequential	**100% ordering agreement (24/24)**					

Table 8: F1 score for failure / predicate detection across three environments. All methods use the F1-optimal threshold; ETL (CP) additionally reports the class-conditional split-CP threshold. Per-column best in **bold**.

Method	STACKING	SORTING	MW-PICK-PLACE	Avg
logpZO	0.844	0.534	**0.992**	0.790
PCA-kmeans	0.852	0.585	0.815	0.751
ETL (best F1)	**0.897**	**0.593**	0.961	**0.817**
ETL (CP)	0.887	0.556	0.923	0.789

Table 9: Target-conditioned atomic similarity versus temporal composition (repeated-split protocol, 20 splits, mean $\pm$ 95% CI where applicable). Both columns use the same encoder, targets, labels, and F1-optimal threshold; only temporal composition differs. Wrong-order detection is the fraction of wrong-order failure episodes on which the monitor raises a violation.

Measure	Atomic (order-blind)	ETL (order-sensitive)
STACKING: generic episode F1	0.918 ± 0.001	0.917 ± 0.004
SORTING: generic episode F1	0.562 ± 0.016	0.595 ± 0.012
STACKING: wrong-order detection	0.577	$\mathbf{0.885}$
SORTING: wrong-order detection	≤ 0.050	$\mathbf{0.827}$

H.3 Sequential Predicate Trace

Figure 3 illustrates dual-predicate traces for the pick-place-wall task. The plots show distances to both subgoal embeddings (z_A for grasp and z_B for place), along with corresponding predicate activations.

The embedding distances to z_A and z_B decrease at distinct timesteps corresponding to grasp and placement events. This temporal separation is preserved in the embedding space, allowing ETL to correctly identify the ordering of subgoals across all trajectories.

I Additional DROID Evaluation Details

I.1 Tabularized Results

We present tabularized results from the experiments from Section 5.3 in Table 10.

I.2 Ground Truth Construction

Unlike simulation environments, DROID does not provide explicit task-phase annotations. We derive ground-truth predicates directly from proprioceptive signals:

$$\pi_{\text{hold}}(t) : \text{gripper}(t) > 0.5$$
$$\pi_{\text{release}}(t) : \text{gripper}(t) < 0.15 \ \wedge \ \exists\, t' < t,\, \pi_{\text{hold}}(t')$$

The sequential specification is defined as:

$$\exists\, t_1 < t_2 : \pi_{\text{hold}}(t_1) \wedge \pi_{\text{release}}(t_2),$$

corresponding to a pick-and-place interaction. For tasks where proprioception is not sufficient to identify the phases of a task (the multi-phase tasks of Table 10), we manually annotate the videos: an annotator marks the

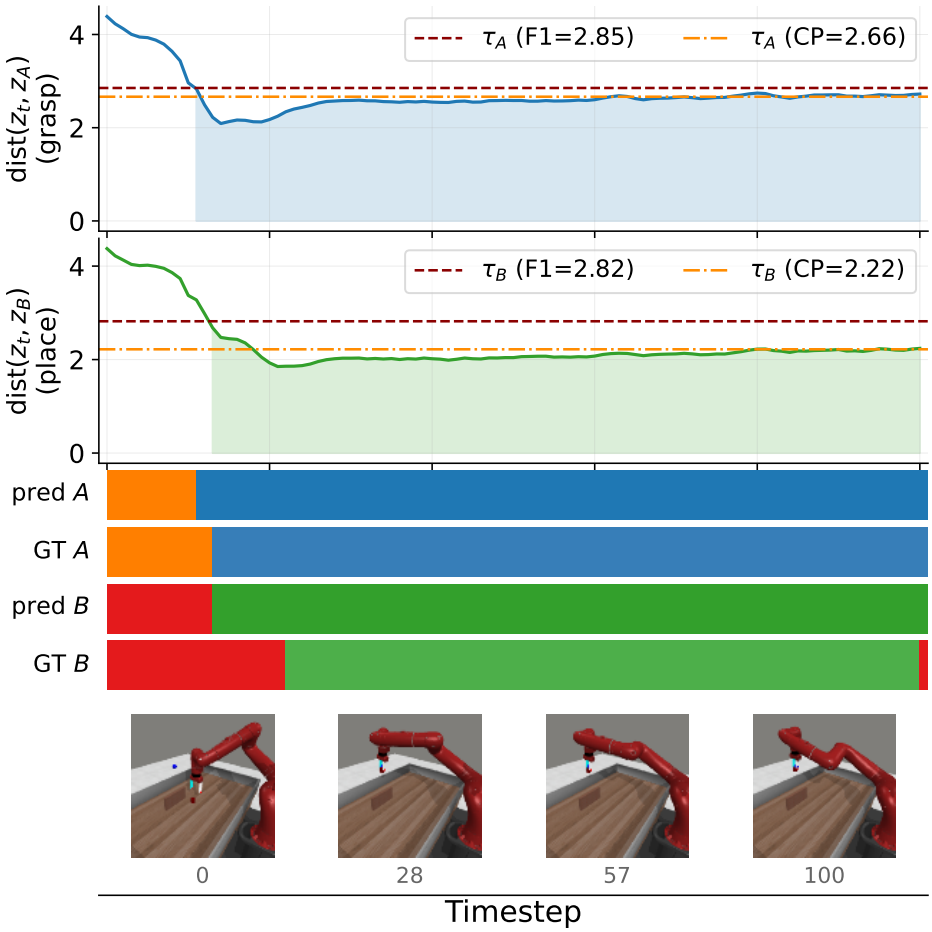


Figure 3: Dual-predicate Boolean timelines for `mw-pick-place-wall`. Top panels show distances to grasp (z_A) and place (z_B) embeddings with thresholds ϵ^* and ϵ_{CP}. Lower panels show ETL and ground-truth predicate activations.

start and end frame of each phase from the task description, and the ground-truth predicate for a phase is positive on frames inside its annotated window and negative on all other frames of the episode.

Splits and target frames All splits are made at the trajectory level: a demonstration contributes frames to exactly one of the reference, calibration, or test sets, so no frame of a test demonstration is used to construct targets or fit thresholds. Target frames for a phase predicate are selected from the reference demonstrations as frames inside the annotated (or proprioceptively derived) window for that phase, and their embeddings form

Table 10: Comparison of ETL (SVD-VAE) and Qwen2-VL-2B on the five phasic DROID tasks. F1 and agreement are frame-level means over each task's held-out demonstrations; Seq. indicates whether the sequential phase ordering was recovered for that task.

		ETL			Qwen2-VL-2B		
Task	**Ph.**	**F1**	**Agr.**	**Seq.**	**F1**	**Agr.**	**Seq.**
Obj. → paper	2	0.968	0.977	✓	0.493	0.383	✓
Tap → spoon	2	0.839	0.937	✗	0.292	0.700	✓
4-way chain	4	0.531	0.862	✓	0.585	0.811	✗
White → blue cloth	3	0.884	0.959	✓	0.335	0.258	✗
Bottles → stove	3	0.844	0.966	✓	0.244	0.685	✗
Mean	–	**0.813**	**0.940**	**4/5**	**0.390**	**0.567**	**2/5**

$\mathcal{Z}_{target}$; thresholds are then fit on the calibration demonstrations and all reported metrics are computed on the test demonstrations.

I.3 VLM Query Template

Concretely, all VLM queries get the fixed system prompt: *"You are evaluating whether a robot arm is currently performing a specific manipulation action. Look carefully at the image and answer only with 'yes' or 'no'."* The query is also appended with per-episode, per-phase user prompts (one question per phase, asked per frame with the frame image attached); e.g., for task instruction "pick up the black object from the striped towel…", we add prompts: *"Is the robot gripper currently closed and holding the black object it picked up from the towel?"* We use Qwen2-VL-2B-Instruct [Wang et al., 2024] with greedy decoding. Each frame of a test demonstration is queried independently (no temporal context) once per phase, so the VLM produces one answer per (frame, phase) pair. The response is lower-cased and stripped, and parsed to `True` if it begins with "yes" and `False` otherwise; any other response is treated as `False`. The resulting per-frame Boolean sequence is scored with the same frame-level F1, agreement, and sequential-ordering metrics as the ETL monitor.

I.4 Encoder and Representation

We use the **SVD VAE** from Ctrl-World [Guo et al., 2026], a video diffusion model trained directly on DROID data. Each frame is encoded into a $4 \times 24 \times 40$ latent tensor, flattened to a 3,840-dimensional vector. Distances are computed using cosine similarity. This encoder is domain-matched to DROID and provides stronger geometric separation than general-purpose encoders such as DINOv2.

We evaluate three frozen general-purpose encoders (DINOv3, DINOv2, and CLIP) against the domain-matched SVD-VAE under the full DROID protocol at the F1-optimal threshold, keeping targets, labels, thresholds, and specifications fixed; per-task results are reported in Table 12.

Can general-purpose encoders reliably monitor temporal ordering? Table 11 compares frozen general-purpose encoders with the domain-matched SVD-VAE on DROID. General-purpose encoders achieve strong frame-level discrimination, with DINOv3 obtaining the highest mean F1 (0.834), followed by DINOv2 (0.817) and SVD-VAE (0.813). However, SVD-VAE correctly recovers temporal ordering in $4/5$ tasks, compared to $2/5$ for DINOv3, $1/5$ for DINOv2, and $0/5$ for CLIP. Notably, on the four-phase chain task, SVD-VAE recovers the correct order despite lower frame-level F1, whereas the general-purpose encoders fail despite higher F1 scores. These results highlight that strong semantic discrimination alone is insufficient for reliable temporal monitoring, with domain-matched representations performing better on complex sequential behaviors (Table 12).

Table 11: Encoder ablation on the five DROID tasks (F1-optimal threshold). Mean frame-level F1 and the number of tasks whose sequential phase ordering is recovered. SVD-VAE is domain-matched; the others are frozen general-purpose image encoders.

Encoder	Mean F1	Correct ordering
SVD-VAE (domain-matched)	0.813	$\mathbf{4/5}$
DINOv3	$\mathbf{0.834}$	$2/5$
DINOv2	0.817	$1/5$
CLIP	0.737	$0/5$

One plausible explanation is that the domain-matched video latent produces sparser, phase-specific detections, whereas the general-purpose image encoders produce additional early detections that can reverse the inferred order. Nevertheless, performance improves across CLIP, DINOv2, and DINOv3 in our experiments, suggesting that newer general-purpose representations are narrowing the gap, although they still trail the domain-matched latent on complex temporal ordering.

I.5 Sequential Evaluation Details

We identify 25 episodes containing valid grasp-then-release transitions, using 10 for calibration and 15 for evaluation.

The release predicate exhibits low precision due to visual ambiguity: the approach phase with an open gripper is visually similar to the post-release phase. Despite this, sequential ordering is correctly identified in all episodes.

Table 12: Per-task evaluation of frozen general-purpose encoders against the domain-matched SVD-VAE on DROID (F1-optimal threshold). ✓indicates that the learned predicates also recover the correct sequential phase ordering for that task.

		SVD-VAE (domain-matched)				DINOv3				DINOv2				CLIP			
Task	**Ph.**	**F1**	**Prec.**	**Rec.**	**Seq.**	**F1**	**Prec.**	**Rec.**	**Seq.**	**F1**	**Prec.**	**Rec.**	**Seq.**	**F1**	**Prec.**	**Rec.**	**Seq.**
Obj. → paper	2	0.97	1.00	0.94	✓	0.92	0.97	0.87	✓	0.95	0.96	0.94	✓	0.83	0.82	0.84	✗
Tap → spoon	2	0.84	0.87	0.85	✗	0.86	0.82	0.91	✗	0.83	0.79	0.90	✗	0.76	0.77	0.76	✗
4-way chain	4	0.53	0.77	0.52	✓	0.81	0.88	0.75	✗	0.79	0.84	0.75	✗	0.75	0.78	0.73	✗
White → blue cloth	3	0.88	0.87	0.90	✓	0.79	0.79	0.82	✗	0.78	0.81	0.78	✗	0.63	0.68	0.59	✗
Bottles → stove	3	0.84	0.99	0.74	✓	0.79	0.76	0.83	✓	0.73	0.70	0.79	✗	0.71	0.70	0.72	✗
Mean		**0.813**	**0.90**	**0.79**	**4/5**	**0.834**	**0.84**	**0.84**	**2/5**	**0.817**	**0.82**	**0.83**	**1/5**	**0.737**	**0.75**	**0.73**	**0/5**

Table 13: Sequential predicate evaluation on DROID (SVD VAE, wrist camera).

Predicate	F1	Precision	Recall	Agreement	Seq. Agreement
π_{hold}	0.666	0.721	0.750	0.701	**1.000**
π_{release}	0.258	0.162	1.000	0.163	

I.6 Discussion of Failure Modes

ETL performance depends on the separability of task-relevant concepts in the embedding space. For manipulation tasks, success states (e.g., button press, object placement) are highly distinctive, leading to sharp transitions in embedding distance and high predicate accuracy.

Failure modes arise when embeddings corresponding to different semantic states overlap in the latent space, which can result in delayed or premature predicate activation near decision boundaries. However, such cases are rare in these tasks, and conformal thresholds help mitigate missed detections by prioritizing recall.

Overall, these qualitative results reinforce that embedding distances provide a reliable and interpretable signal for detecting semantic events and temporal structure in manipulation tasks.

We evaluate whether *Embedding Temporal Logic* (ETL) monitoring can serve as a competitive, interpretable alternative to state of the art failure predictors for generative robot policies. Our central hypothesis is that explicitly encoding the *sequential milestone structure* of a task into a small number of latent spec anchors, and then asking whether each anchor was ever approached during execution, yields a sharper failure signal than methods that either model the full observation distribution or score each timestep independently against a single goal embedding.

I.7 Overhead on Runtime Monitoring

We evaluate the overhead incurred by ETL monitors during deployment for real world systems by computing overhead costs for DROID. For the fragment evaluated in this paper (unbounded **F**, **G**, nested eventualities for sequential patterns, and their Boolean combinations), the monitor output can be updated incrementally without retaining the trace: each temporal operator keeps a constant-size summary (a flag or running extremum for **F**/**G**, a progress index for sequential patterns), so each step costs $O(|\varphi|)$ time and the monitor uses $O(|\varphi|)$ memory, independent of t. This bound is specific to that fragment: time-bounded operators require state proportional to their window width, and arbitrary ETL formulas require a standard automaton construction whose state is independent of trace length but may grow with formula complexity. We additionally ran experiments for an incremental monitor frame-by-frame over all recorded traces and outline the computation times in Table 14. The incremental monitoring computation adds less than 20 μs per frame on CPU beyond the encoder forward pass. Thus, ETL monitoring can be performed at real-time robotic control frequencies.

Table 14: Monitoring overhead costs for ETL monitors

Component	Measured Cost
Atomic predicate evaluation (m=1/10/19)	13.9/16.2/16.6 μs per frame on CPU
Incremental temporal update for Seq. $A \rightarrow B$	0.09 μs per frame on CPU
Encoder forward pass batch size 1, RTX5090	4.69–6.49 ms per frame
End-to-end encoder and monitoring runtime	4.78–6.52 ms per frame / 153–209 Hz

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main contributions and scope are discussed in the last paragraph of Section 1, which links the sections where the contributions are discussed. We scope the paper to the claim and acknowledge limitations (see limitations/next question).

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations of the approach and evaluation are discussed in the context of necessary future work in Section 6.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: There is one theoretical claim made in the paper regarding Conformal Threshold prediction, and a formal theorem and proof are included in Appendix F

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.

- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Artifacts, instructions, and documentation have been included and released for reproducibility at `https://github.com/ETLMonitoringAuthors/ETLMonitoring` and this link is included as a footnote in the first paragraph of Section 5. We include explanations of how to calibrate necessary thresholds and how to utilize work outside of reproducing experiments in Sections 3, 4, and 5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The source code necessary to reproduce the experimental results are provided as supplemental material along with the main paper submission at `https://github.com/ETLMonitoringAuthors/ETLMonitoring` and this link is included as a footnote in the first paragraph of Section 5.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.

- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We include explanations of how to calibrate necessary thresholds and how to utilize work outside of reproducing experiments in Sections 3, 4, and 5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Yes, we perform confidence interval experiments and report the results in Section 5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Computing hardware details are included in Section 5.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The authors have reviewed the Code of Ethics and have ensured conformance with the potential harms and societal impacts sections of the code.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: In Section 1, we indirectly discuss the positive social impacts of the work by discussing how modern autonomous systems require safety verification, and how our work has the ability to ensure improvement of the monitoring abilities. ETL monitors may improve the transparency and safety of perception-based autonomous systems by exposing failures of learned representations during deployment. However, incorrect monitor outputs may create false confidence or unnecessary interventions, especially when embedding spaces fail to separate task-relevant concepts. Deployment should therefore pair ETL monitors with validation under distribution shift and human-interpretable failure analysis.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper does not release data or models that have not been released before; the datasets in the paper are reused from prior works and our contribution is monitoring for safety, liveness, etc. during runtime.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We credit Agrawal et al. [2025] and Hafner et al. [2025] for the Dubins code in Section 5.1, Nicklas Hansen [2026] for Metaworld in Section 5.2, and Khazatsky et al. [2024] for DROID in Section 5.3.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Monitoring datasets have been documented and provided at `https://github.com/ETLMonitoringAuthors/ETLMonitoring` and this link is included as a footnote in the first paragraph of Section 5.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This work does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: This research does not have any human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development of this research did not use LLMs for any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.