LEARNING STACK STABILITY THROUGH PHYSICAL PERTURBATION TESTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Stack stability concerns whether an arrangement of physical objects will remain intact. Physics-based simulations require detailed scene parameters, such as friction coefficients which are often unavailable, while visual classifiers predict stability labels without understanding how failure develops. We propose StackProbe, which learns a state representation from images and available records, by observing how stacks respond to various physical perturbations. Then we can assess stack stability based on the learned state representation. During training, we test the learned state representation under different physical perturbations and refine it using responses from high-fidelity simulations. At inference, we use the learned state representation to predict whether, when, and how a stack may fail, without running a simulator. We evaluate StackProbe on two simulation benchmarks and data from a deployed industrial palletizing system. Empirical results show that StackProbe reduces missed-failure rates by over 32% compared with the strongest learned baseline, while its representation improves average precision for industrial collapse prediction by at least 6.0 percentage points. Code is available at https://anonymous.4open.science/r/StackProbe/.

1 INTRODUCTION

Given a configuration of objects, will it remain intact under the perturbations it may encounter? This question is formalized as stack stability, which is important for various real-world applications such as automated object stacking (Groth et al., 2018a), cargo loading and transportation (Ramos et al., 2017), and tray-based robotic transport (Heins & Schoellig, 2023). These scenarios naturally involve physical perturbations, where stability depends not only on how the objects are arranged, but also on how they are perturbed. Consider a tower of blocks resting on a horizontal tray. The tower may remain intact when the tray accelerates gently, but a stronger acceleration can cause a block to slide, tip, or fall. Whether the tower remains stable therefore depends on both its initial arrangement and the motion of the tray.

Existing methods for assessing stack stability include physics-based models and direct visual classifiers. Traditional **physics-based models** use explicit descriptions of object geometry, contact, and friction to evaluate equilibrium (Mosemann et al., 1997; Liu et al., 2024). Related physics-based approaches further incorporate support motion to assess cargo stability or plan robotic transport (Ramos et al., 2017; Heins & Schoellig, 2023). Although these methods explicitly characterize the mechanics, they require detailed scene parameters that are often unavailable from visual observations. Instead, **visual classifiers** directly learn to map images of object configurations to stability labels (Lerer et al., 2016; Li et al., 2017; Groth et al., 2018a). However, a stability label only summarizes the outcome without describing the physical process that leads to failure. Moreover, configuration-only predictions cannot distinguish the effects of different perturbations.

To model how a stack responds to physical perturbations, we can start from the features learned by video encoders such as V-JEPA (Mur-Labadia et al., 2026). These features capture information about objects, geometry, and motion, providing a basis for representing the stack's current state. The idea is to predict in latent space how this state would evolve under a perturbation, allowing us to anticipate its physical effects before applying it. Such predictions require a representation that retains the relevant physical information as the stack evolves.

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

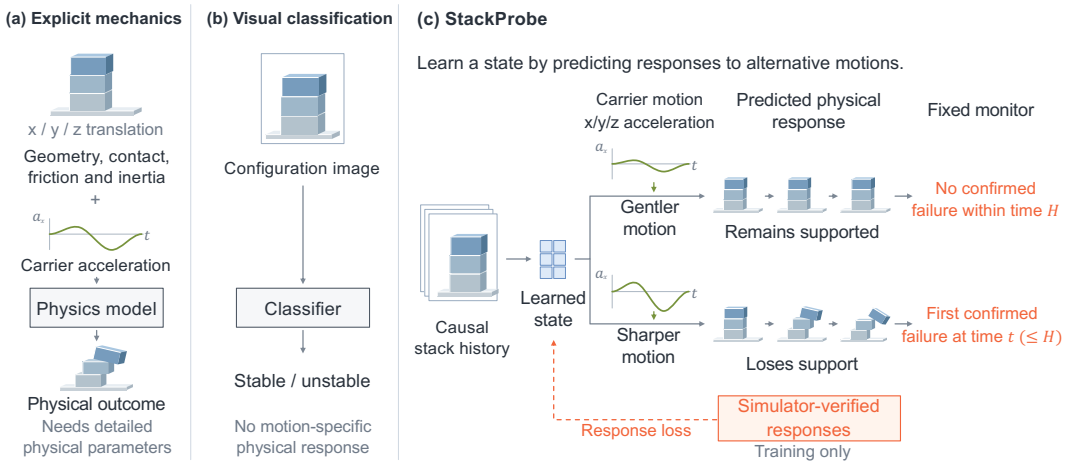


Figure 1: Three views of stack stability under carrier motion. Explicit-mechanics methods predict a physical outcome from detailed scene parameters and a specified motion, while visual classifiers map a configuration image to a motion-agnostic stability label. StackProbe instead learns a state from simulator-verified responses and predicts its motion-conditioned physical response.

In this paper, we propose **StackProbe**, a framework for learning state representations for stack stability through physical perturbation tests. The core idea is to test a learned state by how well it predicts the stack's responses to different perturbations, as illustrated in Figure 1. During training, we test whether this representation remains predictive as the stack evolves. A motion from time t_a to t_c can be divided at an intermediate time t_b: p is the segment from t_a to t_b, and q is the segment from t_b to t_c, so $p \oplus q$ is the original motion. Starting from a state representation derived from a pretrained encoder (Mur-Labadia et al., 2026), the model predicts the stack's response in two ways. One predicts the response to the full motion $p \oplus q$ at once. The other first predicts the state reached after p, then uses that state to predict the response to q without another observation. The two predictions should agree during q, because splitting the motion does not change the underlying physical process. We use their disagreements to select physical tests and supervise both predictions with the same simulated response, since they can agree and still be physically wrong. At inference, StackProbe directly predicts the stack's response to a proposed support motion. The predicted response is then checked against physical criteria to answer whether, when, and how the stack may fail.

We evaluate StackProbe in simulation using stacks from the public PLP and ShapeStacks datasets. StackProbe predicts failures more accurately than visual classifiers and learned response predictors. It reduces the missed-failure rate by over 32% on each benchmark compared with the strongest learned baseline. We also use the learned representation in an online stability assessor deployed on a robotic palletizing line at a logistics facility. Across 1,562 real pallet-building episodes, our method improves average precision for collapse prediction by at least 6.0 percentage points compared with normal visual predictors.

We make three contributions:

(1) Learning through physical tests. We introduce StackProbe, which learns state representations for stack stability by predicting responses to physical perturbations.

(2) Continuation testing. We develop tests that compare full-motion predictions with predictions resumed from an intermediate state. Their disagreements guide simulator tests, and the simulated responses supervise both predictions. Theory explains why these disagreements reveal errors.

(3) Experiments. We demonstrate improved failure prediction on PLP and ShapeStacks. The learned representation also improves collapse prediction in industrial palletizing.

2 Related Work

Stability and interventions. Visual models predict stack stability and motion (Lerer et al., 2016; Groth et al., 2018a), or assess non-disruptive extraction (Ahuja et al., 2020). CoPhy infers hid-

den properties from observed trajectories to predict dynamics after initial-scene interventions (Baradel et al., 2020). Simulation-trained recurrent models assess loading sequences from known box attributes (Mazur et al., 2025); controllers handle nonprehensile transport under inertial uncertainty (Heins & Schoellig, 2025). StackProbe predicts responses to prescribed carrier motions from causal visual evidence.

Predictive states and continuation. DINO-WM predicts action-conditioned features (Zhou et al., 2025); latent overshooting and V-JEPA 2-AC train multistep rollouts (Hafner et al., 2019; Assran et al., 2025). SG-JEPA trains its encoder through gravity-conditioned recursive prediction (Liu et al., 2026). For observable autonomous dynamics, Deep-OSG regularizes direct–composed agreement or supervises both routes with the same observed endpoint (Chen & Wu, 2023). StackProbe tests compressed visual states on alternative continuations without another observation, carrying motion constraints and monitor memory.

Selecting physical tests. Plan2Explore uses ensemble disagreement to plan informative interactions for world-model learning (Sekar et al., 2020). StackProbe compares two routes through one model and tests failure disagreement near the reference plan. Temporal-logic falsification searches for violating inputs (Annpureddy et al., 2011), while reliability analysis measures distance to failure (Hasofer & Lind, 1974). Our radius compares legal carrier motions under a fixed horizon and monitor.

3 Task: Plan-Conditioned Stack Stability

Prior work predicts stability from a fixed scene, models motion from explicit parameters, or intervenes on the initial scene (Lerer et al., 2016; Ramos et al., 2017; Baradel et al., 2020). We instead ask how the observed stack responds to a planned carrier motion, i.e., whether, when, and how it fails, and how far the plan is from failure. This response defines StackProbe's target and tests in Section 4.

Evidence and motion queries. Prediction begins at an observation *anchor* t, with evidence $o_t = (I_{\le t}, r^{\mathrm{obs}}_{\le t}, M)$ containing images, records, and availability/precision masks. A resting anchor uses a synchronized three-view snapshot; a moving anchor uses a causal 16-sample three-view clip. Public conditions η supply calibration, scale, and protocol assumptions. We denote a motion program by q, its acceleration trace by u_q, and the reference plan by $\bar{q}$. A non-learned *registration* λ_t specifies the carrier boundary conditions, time since reset, remaining horizon H_t, and legal motion domain $\mathcal{U}(\lambda_t)$. The reset fixes the reference configuration: it is the settled snapshot for a resting anchor and the first visible clip sample for a moving anchor. A prefix p followed by a continuation q is written $p \oplus q$.

Response and first failure. The model predicts a raw physical response Ψ: poses, velocities, and support/contact quantities over time. A fixed monitor applies physical checks to this response, covering relative configuration, support and landing, and fall and containment. Failure can precede collapse: an object's carrier-relative orientation change θ from reset confirms failure after remaining at least 15° for a prescribed dwell time. Its margin $1 - \theta/15^\circ$ is nonpositive during violation (Appendix B.3). Because a violation may begin before a motion is split, monitor memory μ_t carries unfinished dwell intervals, support references, and confirmed events across the split. It starts with zero accumulated dwell at reset.

For each check j, let τ_j be its query-relative confirmation time, or $+\infty$ if unconfirmed by H_t:

$$\tau = \min_j \tau_j, \qquad Y = \mathbf{1}[\tau \le H_t], \qquad \widetilde{\tau} = \min\{\tau, H_t\}.$$

Thus $Y = 0$ records no confirmed failure within H_t, with the time censored at H_t. When $Y = 1$, tied first modes are the checks confirmed by $\min\{H_t, \tau + \epsilon_{\mathrm{tie}}\}$; otherwise the mode set is empty. The complete task response Φ combines these outputs with the task-supported physical response and masks. Numerical invalidity returns `INVALID`; missing required evidence or registration returns `unavailable`. Neither means no failure. A prefix is *live* when it is legal, numerically valid, and has no failure confirmed by its endpoint; confirmed events are terminal.

The exact task state is therefore $\chi_t = (s_t, \mu_t; \kappa^\star)$: physical state, monitor memory, and query-fixed context. Its response is $\Phi(\chi_t, \lambda_t; q)$; the model estimates this information from available evidence.

Failure radius around a plan. For a non-failing reference plan $\bar{q}$, the failure radius measures the smallest motion change that causes failure. For a fixed registered query, let $\mathcal{F}$ contain legal motions with valid responses and $Y(q) = 1$. For equal horizons H, a dimensionless distance compares full acceleration traces rather than object displacements:

$$d_{\mathrm{act,H}}(q, q') = \left[\frac{1}{H a_{\mathrm{ref}}^2} \int_0^H \|u_q(r) - u_{q'}(r)\|_2^2 \, \mathrm{d}r \right]^{1/2}, \qquad \rho_{\mathrm{fail}}(\bar{q}) = \inf_{q \in \mathcal{F}} d_{\mathrm{act,H}}(q, \bar{q}), \tag{1}$$

where $a_{\mathrm{ref}} > 0$ is fixed and the reference response is valid. For example, distance 0.1 means an RMS acceleration change of $0.1 a_{\mathrm{ref}}$ over the query horizon. The radius is zero for a failing plan and $+\infty$ if no valid legal motion fails. Let $\mathcal{C}$ be a finite legal bank containing $\bar{q}$, and $\widehat{\mathcal{F}}_{\mathcal{C}}$ its valid predicted failures ($\widehat{Y} = 1$). For an available query with a valid predicted reference, we report

$$\widehat{\rho}_{\mathcal{C}}(\bar{q}) = \min_{q \in \widehat{\mathcal{F}}_{\mathcal{C}}} d_{\mathrm{act,H}}(q, \bar{q}) \quad \text{if } \widehat{\mathcal{F}}_{\mathcal{C}} \neq \varnothing; \quad \texttt{not found} \text{ otherwise.} \tag{2}$$

Invalid nonreference candidates are excluded and counted separately. The `not found` outcome means only that the tested bank contains no predicted failure. Within a fixed-horizon pool we write d_{act}; Appendix B specifies the full protocol.

4 StackProbe: Learning a Continuable State Representation

Building on Section 3, StackProbe learns a state that can still predict the stack's response after part of a motion has unfolded. It makes that prediction in two ways, uses their disagreement to choose simulator tests, and learns from the verified response (Figure 2). Unlike prior rollout or consistency objectives (Zhou et al., 2025; Chen & Wu, 2023), this test is tied to physical outcomes and failure.

4.1 State and physical response

The state separates an evolving core c, which must retain information such as velocity and contact state, from fixed context κ and monitor memory μ.

At anchor a, an encoder initialized from V-JEPA 2.1 forms a fused code $z_a = E_\theta(o_a; \eta)$ and a smaller dynamic core $c_a = C_\omega(z_a)$, of dimensions 512 and 256 respectively (Mur-Labadia et al., 2026). The same evidence provides κ_a: object identities, masks, attributes, and reset references. Available records replace estimates field by field. It remains fixed between observations and excludes dynamic fields and unrestricted learned features, so continuation prediction must use the evolving core.

To initialize memory, the model reconstructs and replays the reset-to-anchor physical record; resting anchors have zero-duration records. With $\lambda_{[t_0,a]}$ recording carrier boundary and timing from reset t_0 to a, the full model state is

$$\widehat{\mu}_a = \mathsf{Upd}(\mu_{\mathrm{reset}}, \widehat{y}^{\mathrm{causal}}_{[t_0,a]}; \kappa_a, \lambda_{[t_0,a]}), \qquad \xi_a = (c_a, \widehat{\mu}_a; \kappa_a).$$

For a legal query, the predictor advances the core and a shared reader decodes physical quantities:

$$\widehat{c}_{a+r} = P_\psi(c_a, q_{[0,r]}, r; \kappa_a, \lambda_a), \qquad \widehat{y}_{a+r} = R_\gamma(\widehat{c}_{a+r}; \kappa_a, \lambda_a), \quad 0 \le r \le H_a. \tag{3}$$

Here $q_{[0,r]}$ is the motion prefix through relative time r. Fixed maps assemble these outputs into $\widehat{\Psi}(\xi_a, \lambda_a; q)$; the monitor starts from $\widehat{\mu}_a$ and computes failure, time, and modes. Separate masks track evidence availability, target support, and numerical validity; predicted early failure cannot remove later targets.

4.2 Testing the state on continuations

Fitting complete responses alone need not make an intermediate state reusable, so we test alternative continuations from the same physical state. The simulator executes a live prefix $p : a \to b$ once and saves χ_b. The model advances its core and monitor memory through p without a cut observation:

$$\widehat{c}_b = P_\psi(c_a, p, \Delta; \kappa_a, \lambda_a), \qquad \widehat{\mu}_b = \mathsf{Upd}(\widehat{\mu}_a, \widehat{y}_{[a,b]}; \kappa_a, \lambda_{[a,b]}), \qquad \widehat{\xi}_b = (\widehat{c}_b, \widehat{\mu}_b; \kappa_a).$$

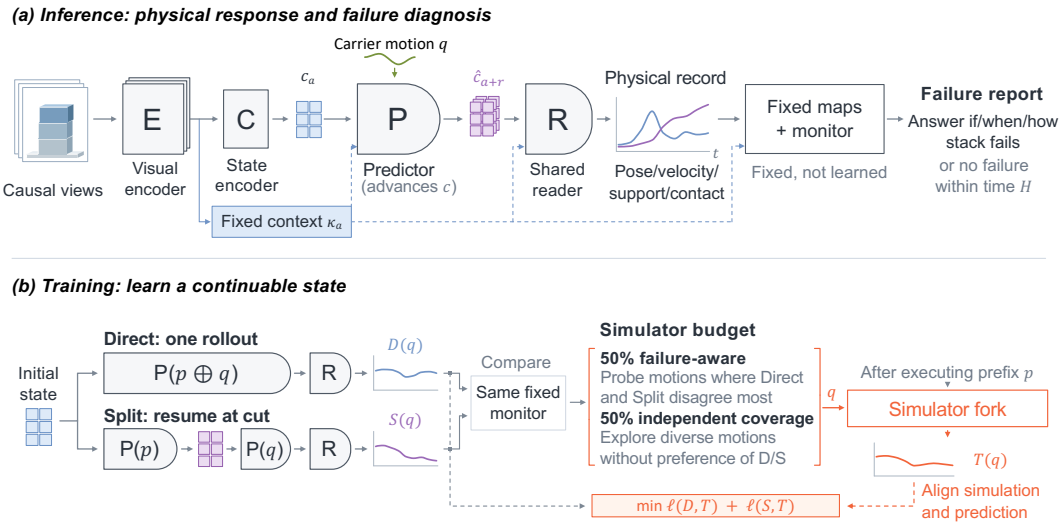


Figure 2: StackProbe inference and training. (a) The learned state predicts a motion-conditioned physical response, which a fixed monitor converts into a failure report. (b) The direct route $D(q)$ retains the response tail of $p \oplus q$; the split route $S(q)$ resumes after p without another observation. Both routes share context and monitor memory and are supervised by the same simulator response $T(q)$. Half the post-seed simulator budget prioritizes failure-bit disagreement, and the other half provides independent plan coverage.

Here $\Delta = b - a$ and $\lambda_b = \mathsf{Res}(\lambda_a, p)$ carries the remaining horizon and original motion constraints. An action-only generator supplies a candidate bank $\mathcal{C}_b$ containing the planned continuation $\bar{q}$. It checks both q and $p \oplus q$ against the inherited motion constraints, without reading simulator outcomes.

The split route resumes each candidate from $\widehat{\xi}_b$; the direct route predicts the full motion from a and retains its raw-response tail:

$$\widehat{\Psi}_b^S(q) = \widehat{\Psi}(\widehat{\xi}_b, \lambda_b; q), \qquad \widehat{\Psi}_b^D(q) = \mathrm{Tail}_p\, \widehat{\Psi}(\xi_a, \lambda_a; p \oplus q). \tag{4}$$

Both routes share the grid, $\widehat{\mu}_b$, κ_a, and fixed monitor. The tail operator crops and reindexes raw samples after b; confirmation times and modes are recomputed from the incoming memory. Direct failure before the cut is recorded separately as a pre-cut false alarm.

4.3 Selecting and Learning from Physical Tests

We prioritize disagreement about failure near the reference plan. Let $\mathcal{C}_b^{\mathrm{mon}}$ contain candidates valid for both routes and live through the direct prefix. The *failure-set* rule chooses the closest to $\bar{q}$ for which the predicted event bits disagree:

$$q_b^\dagger \in \arg \min_{q \in \mathcal{C}_b^{\mathrm{mon}}:\, \widehat{Y}_b^D(q) \neq \widehat{Y}_b^S(q)} d_{\mathrm{act,H_b}}(q, \bar{q}).$$

If the routes agree on every event bit, their physical margins may still differ. The fallback compares the continuous monitor margins available to both routes on $\mathcal{C}_b^{\mathrm{mon}}$ and maximizes their normalized trace difference,

$$\delta_b(q) = \|\widehat{\Psi}_b^S(q) - \widehat{\Psi}_b^D(q)\|_{\mathrm{cont}}. \tag{5}$$

Both routes use margins defined over the full registered horizon. If neither comparison is available, selection falls back to action-only coverage, choosing the candidate farthest from its nearest previously attempted program in the same horizon and family. Coverage also receives half of the post-seed budget independently of disagreement: two routes can agree and still be wrong.

The simulator restores χ_b and executes the selected continuation once. The valid response supplies one shared physical target for both routes and enters replay. Every attempted branch counts against the budget.

Ordinary anchors supervise physical responses, causal records, and estimated context fields. Losses use fixed scales and prediction-independent masks, with separate normalization and equal weights.

A verified continuation gives each physical route half the weight of an ordinary response target. An auxiliary stop-gradient exponential-moving-average (EMA) target supervises latent transitions. Appendix C.2 gives training settings, budgets, and acquisition rules. Inference uses only the encoder, predictor, reader, and fixed monitor.

5 Task State and Continuation Error

Section 4 learns a state by checking whether it supports consistent predictions as a motion unfolds. Here we show why that test matters: the right state preserves every future response relevant to failure, and we characterize when response accuracy carries over to failure boundaries and radii. This connects StackProbe's training signal to its inference-time outputs.

5.1 Future responses define the task state

Following predictive-state and behavioral-equivalence principles (Littman et al., 2001; Givan et al., 2003), two exact states at a shared live registration λ are task-equivalent if every legal continuation produces the same complete response:

$$\chi \sim_\lambda \chi' \iff \Phi(\chi, \lambda; q) = \Phi(\chi', \lambda; q) \quad \forall q \in \mathcal{U}(\lambda). \tag{6}$$

An exact representation cannot merge states from different classes: they disagree on at least one query it must answer. The classes therefore specify the coarsest exact task information, including response-relevant monitor memory. Even identical physical states can require different answers when accumulated dwell differs, explaining the separate monitor memory in Section 4.1. Matching a single observed future does not enforce response equivalence: states that agree on one motion may respond differently to another.

The same response criterion must hold after a cut. After a live prefix $p : a \to b$, the residual registration inherits the original motion constraints and grid:

$$\mathcal{U}(\lambda_b) = \{q : p \oplus q \in \mathcal{U}(\lambda_a)\}, \tag{7}$$

$$\Psi(\chi_b, \lambda_b; q) = \mathrm{Tail}_p\, \Psi(\chi_a, \lambda_a; p \oplus q). \tag{8}$$

The second identity follows from deterministic physical composition, related to the semigroup consistency studied in operator learning (Chen & Wu, 2023). It gives the direct and split routes in Eq. (4) the same physical target. Shared registration and incoming model-side memory ensure that their comparison uses the same query and monitor initialization. Only raw samples and validity are tailed; events are recomputed after advancing monitor memory through p.

5.2 From continuation error to failure decisions

Let $T(q)$, $D(q)$, and $S(q)$ denote simulator, direct, and split records on the common target-supported grid. Each includes incoming memory, context, and validity; predictions share model-side memory and context, while T uses their exact counterparts. Let d_{resp} combine normalized physical, memory, context, and validity differences, ignoring unsupported coordinates (a product pseudometric; Appendix A). Define the response errors $e_R = d_{\mathrm{resp}}(R, T)$ for $R \in \{D, S\}$ and route disagreement $\delta_{\mathrm{cut}} = d_{\mathrm{resp}}(S, D)$. The triangle inequality gives two error bounds

$$e_S \le e_D + \delta_{\mathrm{cut}}, \qquad \max\{e_D, e_S\} \ge \tfrac{1}{2}\delta_{\mathrm{cut}}. \tag{9}$$

The first bound controls split error through direct accuracy and continuation consistency; the second makes disagreement an error witness. This explains the training loop: disagreement selects a test, and the executed response supervises both routes. Agreement can still hide a common error, motivating independent coverage. For valid target and route decisions, event-bit disagreement means one route is wrong.

To connect response error to failure decisions, we summarize the continuous monitor checks by a signed field, using quantitative temporal-logic semantics (Fainekos & Pappas, 2009). Each check has a margin that is positive within its acceptable range and nonpositive on violation. For an instantaneous check, the horizon score g_f^{sc} is its smallest sampled margin. For a dwell check, a low margin at one instant is insufficient: the violation must persist. Its score therefore takes the maximum margin in

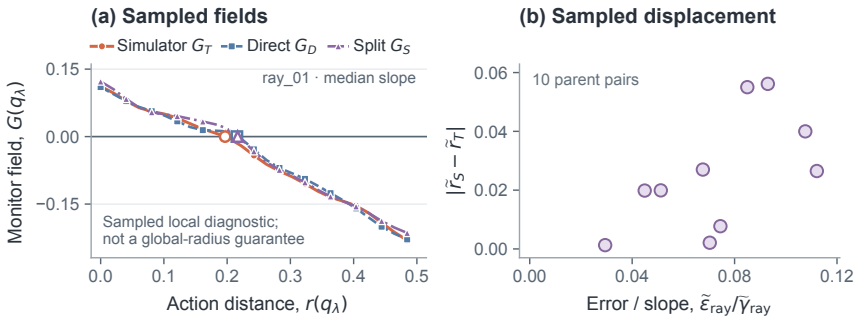


Figure 3: Held-out PLP diagnostic: (a) recorded scores along a pre-specified legal ray; (b) crossing displacement versus sampled error-to-slope ratio for 10 valid pairs. The predicted fields in (a) remain close to the simulator field but cross zero at different action distances.

each completed dwell window, then the infimum over those windows, including inherited samples before a cut. Appendix B.3 specifies the interpolation and sampling convention.

Consider a valid region of motion programs $\mathcal{S}$ (a *stratum*) with fixed object correspondence, active checks, categorical regime, monitor grid, and dwell-window convention. It is *eligible* when its continuous-check set $\mathcal{J}_{\mathcal{S}}$ is nonempty with finite scores, no categorical predicate triggers, and these checks exhaust the event decision. The combined field obeys

$$G(q) = \min_{f \in \mathcal{J}_{\mathcal{S}}} g_f^{\text{sc}}(q), \qquad q \in \mathcal{S}, \tag{10}$$

$$\{q \in \mathcal{S} : G(q) < 0\} \subseteq \mathcal{F} \cap \mathcal{S} \subseteq \{q \in \mathcal{S} : G(q) \le 0\}. \tag{11}$$

A negative field thus implies failure and a positive field excludes it. At zero, the fixed monitor's timing rule decides the event.

Let G_T and G_S be the fields computed from the target and split responses on a shared eligible region. Suppose the field is L_G-Lipschitz under d_{resp} and the uniform bounds are $e_D \le \epsilon_D$ and $\delta_{\text{cut}} \le \epsilon_{\text{cut}}$. Then

$$|G_S(q) - G_T(q)| \le \epsilon_G := L_G(\epsilon_D + \epsilon_{\text{cut}}). \tag{12}$$

Split-route event errors are therefore confined to the boundary band $|G_T(q)| \le \epsilon_G$, including timing ties at zero. Thus small response errors can change decisions near the boundary, motivating failure-focused tests rather than treating every response discrepancy equally. Appendix A gives sufficient metric and score conditions.

5.3 When the failure radius is stable

Near a non-flat crossing, field error displaces the boundary by roughly the error divided by the crossing slope. The following bound formalizes this relation under separation and crossing conditions. Write $r(q) = d_{\text{act}}(q, \bar{q})$, let $\widehat{\mathcal{F}}$ be the valid model-predicted failure set with analysis field $\widehat{G}$ on $\mathcal{S}$, and set $\widehat{\rho} = \inf_{q \in \widehat{\mathcal{F}}} r(q)$.

Proposition 1 (Local failure-radius error)**.** *Suppose the true radius $\rho \in (0, \infty)$ is attained at $q_\star \in \mathcal{S}$, the predicted reference is valid and non-failing, and both responses are valid on $\mathcal{S}$ with the sign relations in Eq. (11).*

For $0 < r_0 \le \rho$ and $m_0, \gamma_G > 0$, assume uniform field error

$$|\widehat{G} - G| \le \epsilon_G < \min\{m_0, \gamma_G r_0\} \quad \text{on } \mathcal{S} \cap \{r \le \rho + r_0\}.$$

On the safe side within $\mathcal{S}$, suppose

$$G(q) \ge \begin{cases} m_0, & r(q) \le \rho - r_0, \\ \gamma_G(\rho - r(q)), & \rho - r_0 < r(q) < \rho. \end{cases}$$

Suppose a legal curve $q_+ : [\rho, \rho + r_0] \to \mathcal{S}$ starts at $q_\star$ and satisfies $r(q_+(s)) = s$ and $G(q_+(s)) \le -\gamma_G(s - \rho)$. If no predicted failure outside $\mathcal{S}$ has radius at most $\rho + r_0$, then

$$|\widehat{\rho} - \rho| \le \epsilon_G / \gamma_G. \tag{13}$$

For a finite bank containing a predicted failure, if its minimum radius obeys $\widehat{\rho}_{\mathcal{C}} \leq \widehat{\rho} + \Delta_{\mathcal{C}}$ *with* $\Delta_{\mathcal{C}} \geq 0$*, then*

$$|\widehat{\rho}_{\mathcal{C}} - \rho| \leq \epsilon_G/\gamma_G + \Delta_{\mathcal{C}}. \tag{14}$$

Accurate responses control boundary displacement through ϵ_G/γ_G; $\Delta_{\mathcal{C}}$ bounds the bank's gap to the nearest predicted failure in the full domain. An action-space coverage measure alone does not establish the latter bound (proof in Appendix A). Accordingly, our radius experiments compare model and simulator on the same candidates, then examine sensitivity to bank size (Appendix C.4).

Figure 3 compares split–simulator crossing displacement with sampled error divided by slope along 10 pre-specified legal PLP rays. It illustrates the local mechanism without establishing the proposition's uniform neighborhood conditions (Appendix A.5).

6 Experiments

We evaluate the claims in this paper by answering (1) whether StackProbe predicts physical response and failure more accurately, (2) whether its state-testing and data-selection mechanisms drive the gains, and (3) whether the learned state supports failure-radius estimation and real-world transfer.

Setup and comparisons. We simulate PLP and ShapeStacks in MuJoCo under fixed density and contact coefficients (Mazur, 2024; Mazur et al., 2027; Groth et al., 2018a; Todorov et al., 2012). Time is normalized by $t_g = \sqrt{L_{\text{ref}}/g}$, where L_{ref} is the square root of carrier footprint area. Equal-input baselines receive the same evidence and motions: a plan-conditioned ShapeStacks classifier, DINO-WM with a physical reader, and Direct (Groth et al., 2018a; Zhou et al., 2025). Direct shares StackProbe's initialization, encoder, reader, and training settings, with non-trunk parameters matched within 5%; a Ramos-style rollout instead receives privileged exact state and physical parameters (Ramos et al., 2017). We use five paired seeds, parent-disjoint splits, and shared test banks; Appendix C gives full protocols, CIs, and counts.

Public-suite results. StackProbe leads the equal-input learners on all four direct-route metrics in both suites (Table 1). Relative to Direct, false-safe falls by 33.4% on PLP and 32.2% on ShapeStacks, with fewer false alarms on both. On PLP, matched coverage lowers false-safe from 11.72% to 10.19%, and acquisition to 7.80%, separating gains from state learning and test selection.

Table 1: Direct-route public-suite results (lower is better). Radius cMAE uses both-found cases, including zero radii. StackProbe uses failure-set acquisition. Bold: best equal-input learner; †: privileged exact-state input; ‡: task adaptation.

Method	nRMSE	False-safe (%)	False-alarm (%)	Radius cMAE
PLP				
Ramos-style mechanics†‡	.067	5.74	4.99	.014
ShapeStacks classifier‡	N/A	20.08	15.94	.036
DINO-WM + reader‡	.122	14.95	12.47	.026
Direct response predictor	.100	11.72	10.90	.020
StackProbe	**.083**	**7.80**	**7.76**	**.014**
ShapeStacks				
Ramos-style mechanics†‡	.078	6.69	5.94	.014
ShapeStacks classifier‡	N/A	16.38	13.36	.036
DINO-WM + reader‡	.109	13.22	11.19	.023
Direct response predictor	.102	12.96	11.81	.019
StackProbe	**.078**	**8.79**	**8.96**	**.013**

On PLP, joint-confirmation time MAE falls from Direct's .171 to .135 t_g and tied-mode macro-F1 rises from .760 to .833; both also improve on ShapeStacks (Appendix C.3).

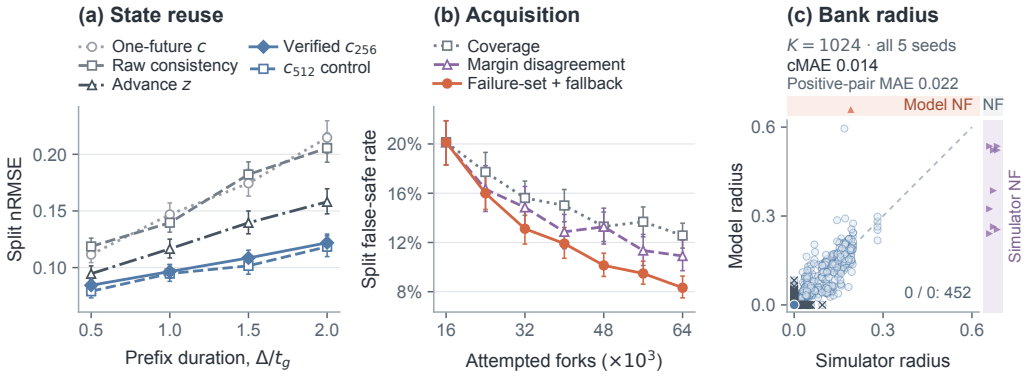


Figure 4: PLP diagnostics: (a) split-response error by prefix for a fixed $2t_g$ continuation; (b) false-safe by attempted forks; (c) predicted versus simulated finite-bank radius. Panels (a)–(b): 95% CIs; NF means `not found`; 0/0 marks 452 cases with both radii zero.

Continuation supervision. On 192 held-out PLP parents, matched controls share simulator forks and direct targets. One-future (one continuation per cut) and raw consistency (latent agreement without split physical targets) both attain direct nRMSE .087 but split nRMSE .162; verified continuations reduce split error to .103, showing that multiple futures make the state more reusable (Figure 4a; Table 2).

Advancing c rather than z reduces split error by another .024; nulling or parent-shuffling c raises it to .266/.219, supporting a dedicated dynamic core.

Acquisition and radius estimates. At 64k attempted calls, failure-set acquisition lowers split false-safe from 12.57% to 8.32%, with paired curve-area and endpoint gains (Table 2; Figure 4b).

On the common 1,024-program bank, PLP radius cMAE falls from Direct's .020 to .014, with a similar ShapeStacks gain (Figure 4c; Appendix C.4).

Table 2: Paired PLP reductions [95% CI]; positive favors StackProbe, and FS area integrates the normalized 16–64k call interval.

Comparison	Metric	Reduction [95% CI]
One-future $c \to$ verified c_{256}	Split nRMSE	.059 [.052, .067]
Raw consistency $\to$ verified c_{256}	Split nRMSE	.059 [.052, .066]
Advance $z \to$ verified c_{256}	Split nRMSE	.024 [.017, .031]
Coverage $\to$ failure-set acquisition	FS curve area (pp)	2.80 [1.87, 3.73]
Coverage $\to$ failure-set acquisition	FS at 64k (pp)	4.24 [3.43, 5.06]

Industrial transfer. At placement 10, five group-disjoint folds evaluate 1,562 deployed palletizing episodes (390 collapses). With the same head, adding frozen StackProbe features to operating records raises AUPRC from .558 to .732, a paired gain of .174 [.139, .201] over records alone, and outperforms frozen Direct, showing transfer to real collapse prediction.

7 Discussion and Conclusion

StackProbe learns a continuable state by selecting physical tests from direct–split disagreement and supervising both routes with simulator responses. Experiments support StackProbe's continuation supervision, failure-aware acquisition, and transfer, while theory gives conditions for accurate failure-radius estimates.

AI use statement. Generative AI assisted literature search, theoretical derivation and proof checks, editing, LaTeX, and plotting code. The authors take responsibility for the final content.

REFERENCES

Sarthak Ahuja, Henny Admoni, and Aaron Steinfeld. Learning vision-based physics intuition models for non-disruptive object extraction. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8161–8168, 2020. doi: 10.1109/IROS45743.2020.9341675.

Yashwanth Annpureddy, Che Liu, Georgios Fainekos, and Sriram Sankaranarayanan. S-TaLiRo: A tool for temporal logic falsification for hybrid systems. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 6605 of *Lecture Notes in Computer Science*, pp. 254–257, 2011. doi: 10.1007/978-3-642-19835-9_21.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew J. Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, et al. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025. URL https://arxiv.org/abs/2506.09985.

Fabien Baradel, Natalia Neverova, Julien Mille, Greg Mori, and Christian Wolf. CoPhy: Counterfactual learning of physical dynamics. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SkeyppEFvS.

Junfeng Chen and Kailiang Wu. Deep-OSG: Deep learning of operators in semigroup. *Journal of Computational Physics*, 493:112498, 2023. doi: 10.1016/j.jcp.2023.112498. URL https://arxiv.org/abs/2302.03358.

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

Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 147(1–2):163–223, 2003. doi: 10.1016/S0004-3702(02)00376-4.

Oliver Groth, Fabian B. Fuchs, Ingmar Posner, and Andrea Vedaldi. ShapeStacks: Learning vision-based physical intuition for generalised object stacking. In *Proceedings of the European Conference on Computer Vision*, pp. 702–717, 2018a.

Oliver Groth, Fabian B. Fuchs, Ingmar Posner, and Andrea Vedaldi. ShapeStacks: Official dataset and code release. GitHub repository, 2018b. URL https://github.com/ogroth/shapestacks. Accessed 19 September 2026.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565, 2019. URL https://proceedings.mlr.press/v97/hafner19a.html.

Abraham M. Hasofer and Niels C. Lind. Exact and invariant second-moment code format. *Journal of the Engineering Mechanics Division*, 100(1):111–121, 1974. doi: 10.1061/JMCEA3.0001848.

Adam Heins and Angela P. Schoellig. Keep it upright: Model predictive control for nonprehensile object transportation with obstacle avoidance on a mobile manipulator. *IEEE Robotics and Automation Letters*, 8(12):7986–7993, 2023. doi: 10.1109/LRA.2023.3324520.

Adam Heins and Angela P. Schoellig. Robust nonprehensile object transportation with uncertain inertial parameters. *IEEE Robotics and Automation Letters*, 10(5):4492–4499, 2025. doi: 10.1109/LRA.2025.3551067. URL https://arxiv.org/abs/2411.07079.

Adam Lerer, Sam Gross, and Rob Fergus. Learning physical intuition of block towers by example. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 430–438, 2016.

Wenbin Li, Aleš Leonardis, and Mario Fritz. Visual stability prediction for robotic manipulation. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2606–2613, 2017. doi: 10.1109/ICRA.2017.7989304.

Michael L. Littman, Richard S. Sutton, and Satinder Singh. Predictive representations of state. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

Andy Zeyi Liu, Haoran Sun, Lucas Baker, Randall Balestriero, and John Sous. Semigroup-JEPA: Latent dynamics consistency for zero-shot physics generalization. *arXiv preprint arXiv:2609.10464*, 2026. URL https://arxiv.org/abs/2609.10464.

Ruixuan Liu, Kangle Deng, Ziwei Wang, and Changliu Liu. StableLego: Stability analysis of block stacking assembly. *IEEE Robotics and Automation Letters*, 9(11):9383–9390, 2024. doi: 10.1109/LRA.2024.3414281.

Philipp Gabriel Mazur. Loading stability benchmark for pallet loading problems. Zenodo, 2024. URL https://zenodo.org/records/13925610.

Philipp Gabriel Mazur, Johannes Werner Melsbach, and Detlef Schoder. Physical question, virtual answer: Optimized real-time physical simulations and physics-informed learning approaches for cargo loading stability. *Operations Research Perspectives*, 14:100329, 2025. doi: 10.1016/j.orp.2025.100329. URL https://kups.ub.uni-koeln.de/79189/1/1-s2.0-S2214716025000053-main.pdf.

Philipp Gabriel Mazur, Frederick Christian Gamer, António G. Ramos, and Detlef Schoder. Standing on a common ground: A comparison of static stability approaches for pallet loading. *International Transactions in Operational Research*, 34(1):501–527, 2027. doi: 10.1111/itor.70059. First published online 21 June 2025.

Meta FAIR. V-JEPA 2 and 2.1: Official PyTorch code and model releases. GitHub repository, 2026. URL https://github.com/facebookresearch/vjepa2. Accessed 19 September 2026.

Heiko Mosemann, Frank Röhrdanz, and Friedrich M. Wahl. Stability analysis of assemblies considering friction. *IEEE Transactions on Robotics and Automation*, 13(6):805–813, 1997. doi: 10.1109/70.650159.

MuJoCo Developers. MuJoCo documentation: State and control, 2026. URL https://mujoco.readthedocs.io/en/stable/programming/simulation.html#state-and-control. Official documentation, accessed 17 September 2026.

Lorenzo Mur-Labadia, Matthew Muckley, Amir Bar, Mido Assran, Koustuv Sinha, Mike Rabbat, Yann LeCun, Nicolas Ballas, and Adrien Bardes. V-JEPA 2.1: Unlocking dense features in video self-supervised learning. *arXiv preprint arXiv:2603.14482*, 2026. URL https://arxiv.org/abs/2603.14482.

António Galrão Ramos, João Tiago Pinheiro Neto Jacob, Jorge Fonseca Justo, José Fernando Oliveira, Rui Rodrigues, and A. Miguel Gomes. Cargo dynamic stability in the container loading problem: A physics simulation tool approach. *International Journal of Simulation and Process Modelling*, 12(1):29–41, 2017. doi: 10.1504/IJSPM.2017.082788.

Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8583–8592, 2020. URL https://proceedings.mlr.press/v119/sekar20a.html.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 79115–79135, 2025.

Appendix A gives the proofs and local diagnostic, Appendix B specifies the observation and simulation protocol, and Appendix C presents experimental details and additional results.

A Proofs and Conditions for the Continuation Analysis

We give the proofs and monitor conditions for Section 5, followed by the protocol for the sampled-ray diagnostic.

A.1 Task partition and registered cuts

At fixed registration λ, Eq. (6) compares the response functions $q \mapsto \Phi(\chi, \lambda; q)$ and is therefore an equivalence relation. If an exact representation assigns the same code to χ and χ', its answers agree for every legal continuation, so $\chi \sim_\lambda \chi'$. Conversely, the equivalence class determines every such answer by definition. The classes therefore form the coarsest exact partition at λ; neither finite-dimensional coordinates nor learned recovery of these classes is assumed.

Monitor memory can distinguish states with identical physical configurations. For example, suppose two states have the same negative margin, but one has accumulated $\Delta_f - \varepsilon$ of a required dwell interval while the other has just entered violation, with $0 < \varepsilon < \Delta_f$. Holding the margin negative for another ε confirms failure only for the first state, so their task responses differ.

Now execute a live prefix $p : a \to b$ from (χ_a, λ_a). The residual domain retains the original travel envelope, grid phase, and join constraints, giving Eq. (7). Determinism makes execution of q from the exact successor (χ_b, λ_b) identical to the suffix of $p \oplus q$ from the initial state. Cropping and reindexing the raw trajectory, including numerical-validity coordinates, proves Eq. (8). Memory advances through p by Eq. (18); the residual support mask, first-passage outputs, and analysis scores are then recomputed from that memory and the suffix. Neither a first-confirmation output nor an aggregate horizon score is obtained by cropping its full-program counterpart.

Retaining the original travel origin is necessary even for a zero-acceleration program. For example, a carrier moving at $0.2L_{\mathrm{ref}}/t_g$ travels $0.4L_{\mathrm{ref}}$ in each of two $2t_g$ segments but $0.8L_{\mathrm{ref}}$ over their concatenation. Testing each segment against a new $0.5L_{\mathrm{ref}}$ travel envelope would admit a program that violates the original constraint; the inherited domain rejects it.

A.2 The monitor field and response error

The field uses the interpolation-at-entry and sampling-during-dwell convention of the continuous checks in Appendix B.3. This shared convention gives the sign relations and response-to-field error bound below.

Sign relations. Consider a persistent check and the fixed family $F_{f,a}$ in Eq. (23). If $g_f^{\mathrm{sc}} < 0$, some admissible a has $F_{f,a} < 0$. Its interpolated margin is negative at a, and every subsequent checked sample through $a + \Delta_f$ is negative. Let $\sigma \le a$ be the start of the violation run containing a; no positive sample occurs between σ and a or before $\sigma + \Delta_f \le a + \Delta_f$. The monitor therefore confirms that run within the horizon. A live incoming state excludes any confirmation before the current query.

Conversely, if the check confirms a run beginning at σ, then σ is an admissible window start, its entry margin is nonpositive, and all samples checked through $\sigma + \Delta_f$ are nonpositive. Thus $F_{f,\sigma} \le 0$ and $g_f^{\mathrm{sc}} \le 0$. For an instantaneous check, the same implications follow directly from its minimum sampled margin. Taking the minimum over the fixed active continuous checks proves Eq. (11) when those checks exhaust the event decision.

An infimum of zero need not confirm failure. With grid spacing $0.025t_g$, dwell $0.25t_g$, and horizon $0.275t_g$, take margin samples $(1, 0, -1, \ldots, -1, 1)$, with the last sample at the horizon. As $a \uparrow 0.025t_g$, $F_{f,a} \downarrow 0$; at that start, the positive endpoint enters the window and cancels confirmation.

Lemma (response-to-field transfer). Fix a legal continuation, registration, and eligible region, with common active checks and window families for the true and predicted responses. Let ϵ_s bound the dynamic-field error on the monitor grid and let ϵ_κ and ϵ_μ bound context and incoming-memory error in fixed normalized metrics. Assume margin maps are Lipschitz in the dynamic fields and context with constants L_s, L_κ, and reconstruction of the retained incoming margin record has error at most $L_\mu\epsilon_\mu$ under the same context. Then

$$|\widehat{G}(q) - G(q)| \le L_s\epsilon_s + L_\kappa\epsilon_\kappa + L_\mu\epsilon_\mu. \tag{15}$$

Proof. The margin maps bound the future sampled-margin error by $L_s\epsilon_s + L_\kappa\epsilon_\kappa$. Including the retained incoming record bounds the error on the combined sample sequence by the right-hand side of Eq. (15). Linear interpolation is a convex combination of adjacent samples, so it cannot increase their maximum error. For each fixed a, the maximum defining $F_{f,a}$ is likewise nonexpansive in the uniform norm. If two bounded families obey $|\widehat{F}_{f,a} - F_{f,a}| \le \varepsilon$ for every a, then

$$\inf_a F_{f,a} - \varepsilon \le \inf_a \widehat{F}_{f,a} \le \inf_a F_{f,a} + \varepsilon,$$

so taking the infimum preserves the bound without requiring an attaining window. The final minimum over the common active-check set is also nonexpansive, proving the result. With identical incoming memory, its error term vanishes.

These conditions concern the whole monitor-required record, including samples before a cut when they complete a dwell window. They do not assert continuity across a categorical switch or a change of object correspondence. First-confirmation time and tied modes require additional timing and ordering conditions and are not bounded by Eq. (15).

A.3 Continuation-cut error and event transfer

For a registered continuation q, write each comparison record as $\mathcal{R} = (\Psi, \mu, \kappa, V^{\mathrm{mon}}, V^{\mathrm{num}})$. The simulator record is $T(q) = (\Psi_T, \mu_b, \kappa^\star, V^{\mathrm{mon}}, V_T^{\mathrm{num}})$. The direct and split records use their respective raw predictions and numerical-validity coordinates, with shared model-side memory $\widehat{\mu}_b$ and context κ_a. All three use the same task-defined support, which includes the monitor-required window and cannot be shortened by a route's predicted stopping time.

Choose fixed scales for supported continuous coordinates, Hamming distances for categorical and validity coordinates, and fixed metrics for memory and context. Their product defines d_{resp}; masked coordinates contribute zero, making it a pseudometric on padded records. This analytical distance is separate from the acquisition score and training loss. At each q, the triangle inequality through D gives

$$d_{\mathrm{resp}}(S,T) \le d_{\mathrm{resp}}(S,D) + d_{\mathrm{resp}}(D,T),$$

while the inequality through T gives $d_{\mathrm{resp}}(S,D) \le d_{\mathrm{resp}}(S,T) + d_{\mathrm{resp}}(D,T)$. These prove both parts of Eq. (9). In particular, agreement alone places no upper bound on error: both routes can produce the same incorrect response.

Suppose the true and split responses lie in a shared eligible region and the score is L_G-Lipschitz under this metric. Then

$$|G_S(q) - G_T(q)| \le L_G d_{\mathrm{resp}}(S(q), T(q)) \le L_G\big(e_D(q) + \delta_{\mathrm{cut}}(q)\big),$$

which yields Eq. (12) under the stated uniform bounds. If $G_T(q) > \epsilon_G$, both fields are positive and both monitor decisions are non-failing. If $G_T(q) < -\epsilon_G$, both fields are negative and both decisions are failing. Thus any event error lies in $|G_T(q)| \le \epsilon_G$, including possible differences at zero. For categorical as well as continuous failures, disagreement between two valid binary predictions still means that exactly one differs from the valid true bit; it does not identify that route.

A.4 Proof of the local radius bound

Write $r(q) = d_{\mathrm{act}}(q, \bar{q})$ and let $h = \epsilon_G/\gamma_G < r_0$. For $q \in \mathcal{S}$ with $r(q) < \rho - h$, inner separation gives $G(q) \ge m_0 > \epsilon_G$ or $G(q) \ge \gamma_G(\rho - r(q)) > \epsilon_G$. Thus $\widehat{G}(q) > 0$ excludes a predicted failure; the condition outside $\mathcal{S}$ excludes the remaining programs in this ball, giving $\widehat{\rho} \ge \rho - h$.

For the upper bound, choose any $0 < \delta < r_0 - h$ and evaluate the legal outward curve at $s = \rho + h + \delta$. The true field is at most $-\epsilon_G - \gamma_G\delta$, hence $\widehat{G}(q_+(s)) \le -\gamma_G\delta < 0$. A predicted failure therefore exists at radius s, regardless of the monitor's decision at zero. Since this holds for every such δ, taking $\delta \downarrow 0$ gives $\widehat{\rho} \le \rho + h$ and proves Eq. (13).

Restricting the predicted failure search to a finite bank gives $\widehat{\rho}_{\mathcal{C}} \ge \widehat{\rho}$. Its candidate-gap premise supplies the other side, so

$$\rho - \epsilon_G/\gamma_G \le \widehat{\rho}_{\mathcal{C}} \le \rho + \epsilon_G/\gamma_G + \Delta_{\mathcal{C}},$$

which proves Eq. (14). A bank without a predicted failing member reports `not found`, and this finite-bank conclusion does not apply.

The assumptions exclude a grazing crossing and an uncontrolled closer failure outside $\mathcal{S}$. When several boundary points attain the true radius, inner separation must cover all closer programs, but an outward curve from one minimizer suffices. An ordinary action-space fill distance does not supply the candidate-gap premise without additional sign and regularity information.

A.5 Sampled-ray crossing diagnostic

For Figure 3, let $\widetilde{G}_R(q_{\lambda_j})$ denote the recorded fixed-horizon monitor score at sampled program q_{λ_j} for the simulator ($R = T$), direct route ($R = D$), or split route ($R = S$); the figure labels these scores G_R. For each route, $\widetilde{r}_R$ is the linearly interpolated first sampled positive-to-nonpositive crossing within a fixed stratum, when one exists. On the prescribed local sample window, define

$$\widetilde{\epsilon}_{\mathrm{ray}} = \max_j |\widetilde{G}_S(q_{\lambda_j}) - \widetilde{G}_T(q_{\lambda_j})|, \qquad \widetilde{\gamma}_{\mathrm{ray}} = \min_j \frac{\widetilde{G}_T(q_{\lambda_j}) - \widetilde{G}_T(q_{\lambda_{j+1}})}{r(q_{\lambda_{j+1}}) - r(q_{\lambda_j})}, \tag{16}$$

where the minimum uses adjacent samples that pass the fixed-stratum and positive-descent screen. The diagnostic compares $|\widetilde{r}_S - \widetilde{r}_T|$ with $\widetilde{\epsilon}_{\mathrm{ray}}/\widetilde{\gamma}_{\mathrm{ray}}$ for the same sampled programs.

These evaluation rays are disjoint from training, continuation evaluation, and the radius bank. Development data fix the direction, scan limit, legality and stratum rules, local window, slope threshold, cut schedule, and seed before held-out evaluation. Parents follow seed order with fixed anchors and plans, and unusable cases remain logged. The initial cohort of 16 expands once to 32 only if fewer than eight paired split crossings survive. Of 16 attempted parents, 12 pass screening and 10 yield paired crossings; two are invalid or switch stratum, and the maximum sample spacing is .03223 in normalized action distance.

Along $q_\lambda = \mathcal{G}(\bar{\alpha} + \lambda v)$, 25 equally spaced samples and at most seven fixed-rule refinements follow the first simulator sign-change bracket. All calls are charged, and the three routes use identical programs without giving either prediction route the true cut state, memory, or outcome. The local window supplies the minimum positive adjacent descent quotient and the maximum split–simulator score discrepancy over common valid samples from the reference through the short outward window. A screen pass requires a safe reference, legal ray and joins, a finite true score with a unique non-flat first crossing, and a fixed stratum and active check; a paired case also requires a valid split crossing there. Reference failure, no sampled crossing, and invalidity remain separate statuses.

Panel (b) includes all finite paired cases passing those rules. The left-panel case has the screened simulator slope closest to the median, with seed-order ties. Together, the panels illustrate how differences between predicted and simulated monitor fields shift the estimated failure crossing along sampled legal motions.

B Observation and Simulation Protocol

The protocol fixes what the model observes, which carrier motions it may query, how failure is confirmed, and how simulator branches are restored.

B.1 EVIDENCE, REGISTRATION, AND ACCESS

At query time t, the model receives $o_t = (I_{\le t}, r^{\mathrm{obs}}_{\le t}, M)$ and declared global conditions η. The evidence contains only information available by t; M records field presence and precision, and simulator state s_t remains a target and scoring object rather than an inference input. Camera calibration, metric scale, protocol constants, and global material-law assumptions belong to η; episode-specific object values belong to context κ.

Registration keeps the motion constraints and monitor clock consistent across a query and its continuations:

$$\lambda_t = (\beta^{\mathrm{car}}_t, e_t, H_t, \zeta_t), \tag{17}$$

where β^{car}_t is the known carrier boundary condition, e_t is elapsed time from the registered reset, $H_t > 0$ is the remaining query horizon, and ζ_t records the spline grammar, grid phase, and admissible coefficient domain. It fixes the legal domain, time origin, output grid, and applicable metric outside the learned state. The boundary β^{car}_t contains the carrier position, velocity, acceleration, and jerk required to test a legal join; carrier orientation and the inertial coordinate frame are fixed.

Every live query starts from a registered reset with a known unconfirmed monitor state and zero accumulated dwell. The reset is the settled snapshot for a resting anchor and the first of 16 synchronized samples in the preceding-t_g clip for a moving anchor. All reset-relative poses, support references, and monitor clocks refer to that same registered instant. The physical history used to generate a moving case can begin earlier; it does not change the current query's reset. Replaying the predicted reset-to-anchor record through the fixed monitor reconstructs $\widehat{\mu}_t$ without simulator-only state or future observation. A moving anchor is `unavailable` if any sample or its registration is missing. If replay confirms failure by the anchor, the protocol returns the terminal event, time, and modes and admits no continuation or radius query; otherwise, a valid replay produces a live anchor. After the anchor, a legal prefix advances memory through its predicted monitor-sufficient record rather than through later observation. A continuation cut preserves the reset and its references instead of starting the monitor again. For a segment from a to b, its monitor-sufficient physical record $y^{\mathrm{mon}}_{[a,b]}$ updates memory by

$$\mu_b = \mathsf{Upd}(\mu_a, y^{\mathrm{mon}}_{[a,b]}; \kappa^\star, \lambda_{[a,b]}), \tag{18}$$

where $\lambda_{[a,b]}$ records the segment's registered boundary and timing, and the predicted route uses its own physical record and context in the same map. If the reset, causal record, or any component of λ_t is missing, the query returns `unavailable` rather than using a learned initialization; target-side μ_t is reserved for labels and teacher-forced diagnostics.

Context κ contains query-fixed identities, masks, geometry and weight, intended supports, and reset references. Reliable records replace estimates field by field, and only estimated fields receive supervision. Dynamic fields, residuals of z_t, and unrestricted learned features cannot enter context.

For the public suites, object mass, center of mass, and inertia follow the fixed-density, centered-CoM reconstruction in Appendix B.4; released mass and CoM are not episode inputs. Later placements start new queries and may change the roster; their order, time, and realized poses remain in $r^{\mathrm{obs}}_{\le t}$.

The reader takes a predicted or encoded core c and κ, but not z_t, exact context $\kappa^\star$, true simulator geometry, or an initial-state shortcut, and predicts the monitor's named metric fields. Fixed maps convert the record into margins and hybrid predicates; only the fixed monitor derives event, time, and mode. Exact s_t, μ_t, and $\kappa^\star$ are reserved for targets, restoration, and explicitly marked diagnostics.

B.2 LEGAL CARRIER-MOTION GENERATOR

The legal trace domain $\mathcal{U}(\lambda_t)$ contains finite piecewise-cubic acceleration programs on the knot schedules registered by ζ_t. Each segment has a finite coefficient vector, every registered join matches acceleration and jerk, and the full trace must satisfy the feasibility constraints below. Ordinary one-segment candidates use six control points per translation axis; a concatenation $p \oplus q$ may retain the registered join between two such segments. The acceleration trace u_q commands carrier translation in the fixed inertial frame and excludes gravity. For carrier footprint area A_{car} and gravitational acceleration magnitude g, the protocol scales are

$$L_{\mathrm{ref}} = \sqrt{A_{\mathrm{car}}}, \qquad t_g = \sqrt{L_{\mathrm{ref}}/g}, \qquad a_{\mathrm{ref}} = g. \tag{19}$$

The $4t_g$ horizon uses $0.0025t_g$ physics steps and $0.025t_g$ response and monitor steps.

The one-segment generator uses a clamped cubic B-spline with six uniformly spaced control abscissae per translation axis and an open-uniform knot vector. Initial acceleration and jerk match the query boundary; terminal acceleration and jerk are zero. A fixed linear projection imposes these four endpoint equalities before a scrambled Sobol sequence with seed 1701 proposes the remaining coefficients. Let

$$v_q(r) = v_q(0) + \int_0^r u_q(s)\,\mathrm{d}s, \qquad x_q(r) = x_q(0) + \int_0^r v_q(s)\,\mathrm{d}s,$$

where $v_q(0)$ and $x_q(0)$ come from β_t^{car}. The travel envelope is centered at $x_{\mathrm{orig}}^{\mathrm{car}}$, the carrier position at the initial query anchor; this origin is retained in the registered domain after every cut. Let $\mathcal{T}_{\mathrm{leg}}$ be the union of all spline knots and the fixed $0.0025t_g$ physics grid. Rejection enforces

$$\begin{aligned} &\max_{r\in\mathcal{T}_{\mathrm{leg}}} \|u_q(r)\|_2 \le 0.75g, && \max_{r\in\mathcal{T}_{\mathrm{leg}}} \|\dot{u}_q(r)\|_2 \le 3g/t_g, \\ &\max_{r\in\mathcal{T}_{\mathrm{leg}}} \|v_q(r)\|_2 \le 0.75L_{\mathrm{ref}}/t_g, && \max_{r\in\mathcal{T}_{\mathrm{leg}}} \|x_q(r) - x_{\mathrm{orig}}^{\mathrm{car}}\|_2 \le 0.5L_{\mathrm{ref}}. \end{aligned} \tag{20}$$

These are registered-grid legality checks, not continuous-time certificates between checked points. Smooth pulses, sinusoids, and sequential two-axis compositions are proposal families for training; chirps and simultaneous three-axis compositions are reserved for held-out-family evaluation. Family templates are evaluated at 64 uniform times and fit by constrained least squares to the spline before legality rejection. Proposed amplitude satisfies $A/g \in [0.1, 0.75]$. A pulse has center $r_c/H \in [0.25, 0.75]$ and Gaussian width $\sigma/H \in [0.08, 0.20]$ on one signed axis. A sinusoid has dimensionless frequency $\nu H \in [0.5, 2]$ and phase in $[0, 2\pi)$. A sequential composition uses two distinct axes with pulse centers separated by at least $0.2H$. Held-out chirps use $\nu_0 H \in [0.25, 1]$ and $\nu_1 H \in [2, 4]$, while the three-axis family sums three simultaneous sinusoidal templates. Sobol coordinates choose intervals, axes, signs, phases, and amplitudes before the common boundary projection and rejection. No exact program is reused across acquisition and evaluation.

Every $q \in \mathcal{U}(\lambda_t)$ is defined on $[0, H_t]$ and satisfies the acceleration, jerk, velocity, travel, and join constraints above. For a prefix $p : a \to b$ of duration $\Delta = b - a$, the residual registration is

$$\lambda_b = \mathsf{Res}(\lambda_a, p) = (\beta_b^{\mathrm{car}}, e_a + \Delta, H_a - \Delta, \zeta_b), \tag{21}$$

Here β_b^{car} is the prefix endpoint, and ζ_b inherits the grid phase and feasibility domain together with the suffix grammar and remaining knots.

The residual domain contains exactly the continuations whose concatenation with p satisfies the original constraints, including travel measured from $x_{\mathrm{orig}}^{\mathrm{car}}$; this gives Eq. (7). Both q under the inherited domain and $p \oplus q$ under the original domain must pass the legality checks. Proposal families need not be closed under cuts. Feasibility constrains the legal domain; the action metric remains the normalized acceleration distance in Section 3. The fixed-$2t_g$ diagnostic in Figure 4(a) uses initial horizon $\Delta + 2t_g$ for $\Delta \in \{0.5, 1, 1.5, 2\}t_g$.

B.3 Fixed monitor and physical records

For a continuous response y_j with nominal lower and upper bounds $\underline{b}_j$ and $\overline{b}_j$, each finite endpoint defines a normalized signed margin,

$$m_{j,-}(r) = \frac{y_j(r) - \underline{b}_j}{\ell_j}, \qquad m_{j,+}(r) = \frac{\overline{b}_j - y_j(r)}{\ell_j}, \tag{22}$$

where $\ell_j > 0$, positive values are acceptable, and a value at or below zero is a violation.

At reset, the intended-support graph considers a directed edge $i \to k$ from object i to an object or the carrier k for every pair in contact. The edge is present exactly when the summed normal-force component on i opposing gravity is strictly positive. This graph is frozen for the query; the active load-bearing graph applies the same positive-force rule at each monitor sample. Let $P_i(r)$ be object i's footprint projected into the carrier plane at query-relative time r, and let P_i^0 denote its footprint at the registered reset. Let S_i^0 be the overlap of P_i^0 with the union of the projected top footprints of its intended supports. We set $w_{s,i} = \sqrt{\mathrm{area}(S_i^0)}$ and let $h_{o,i}$ be the reset-time vertical extent of object i;

a nonpositive area, height, or object weight makes the anchor `unavailable`. Below, w_s and h_o denote the values for the object whose check is being evaluated.

For object i, carrier-relative shift is the planar displacement of its CoM from reset, orientation error is the geodesic change in its carrier-relative orientation, and intended-footprint overlap is $\mathrm{area}(P_i(r) \cap S_i^0)/\,\mathrm{area}(P_i(r))$. Boundary clearance is the signed planar distance from the projected CoM to the fixed carrier footprint, positive inside, and downward displacement is the reset height minus the current CoM height. An intended support path exists when i reaches the carrier through edges present in both the frozen intended graph and the current active graph; a carrier-connected path uses the current active graph alone. An unintended load-bearing landing is an active edge $i \to k$ absent from the frozen intended graph. Each object and each unintended edge has its own check, and the global confirmation time is the minimum over those checks. The record retains tied check identities, which map deterministically to the three failure families in Table 3.

Table 3: Failure families and fixed confirmation rules. Lengths are normalized by the intended-support width w_s or object height h_o.

Family	Physical checks	Confirmation
Relative configuration	Carrier-relative shift $\geq 0.10w_s$, orientation error $\geq 15^\circ$, or intended-footprint overlap ≤ 0.50	Violation persists for $0.25t_g$.
Support and landing	Loss of the registered intended support path or a new unintended load-bearing landing	Predicate persists for $0.25t_g$.
Fall and containment	Downward CoM displacement $\geq 0.50h_o$, loss of every carrier-connected support path, or projected CoM on or outside the carrier footprint	The first two conditions are absorbing; footprint violation persists for $0.25t_g$.

The fixed normalizer is w_s for planar lengths, h_o for vertical displacement, 15° for orientation, object weight for forces, and one for overlap fractions or other dimensionless scores.

A persistent continuous check starts at the entry crossing, linearly interpolated between adjacent samples of the $0.025t_g$ monitor grid. The run confirms at crossing time plus Δ_f if every subsequent grid sample up to and including that time remains nonpositive; otherwise it ends at the first positive sample. A violation already present at reset starts its run there, and a positive sample exactly at a proposed confirmation time cancels the run.

For a persistent categorical predicate, the first violating grid time starts the run, which confirms at the first grid time at least Δ_f later if every intervening sample remains violating. An instantaneous or absorbing check confirms at its first violating grid sample, and an absorbing check then remains latched. The $0.0025t_g$ physics step advances simulator dynamics but is not a separate label grid, so these rules make labels reproducible without claiming continuous-time certification.

An analysis field for the fixed monitor. At a cut, μ_t retains the preceding margin trace or an exact sufficient summary, active relations, and confirmed bits. Append the new margin samples to this incoming record and denote their piecewise-linear interpolation by $\widetilde{m}_f$. For a persistent continuous check with dwell Δ_f, let $A_f(h) = \{a \geq -e_t : 0 \leq a + \Delta_f \leq h\}$ index the windows that start after reset and finish within the residual query. Using the inherited monitor grid $\mathcal{T}$, define

$$\begin{aligned} F_{f,a}(q) &= \max(\{\widetilde{m}_f(a; q)\} \cup \{m_f(t_j; q) : t_j \in \mathcal{T},\ a < t_j \leq a + \Delta_f\}), \\ g_f(h; \chi_t, \lambda_t, q) &= \inf_{a \in A_f(h)} F_{f,a}(q). \end{aligned} \tag{23}$$

Only the window start is interpolated; the remaining terms use exactly the samples checked before confirmation. The window family depends on registration and dwell, not on the response values or their crossing times. An empty family has score $+\infty$; an instantaneous check instead uses the minimum margin over its monitor samples through h. These scores leave the deployed predictor and monitor unchanged; at the fixed horizon, we use

$$g_f^{\mathrm{sc}}(\chi_t, \lambda_t; q) = g_f(H_t; \chi_t, \lambda_t, q). \tag{24}$$

A fixed hybrid-and-monitor stratum holds object correspondence, active continuous checks, categorical regime, window convention, and scoring grid fixed. It is eligible when its nonempty continuous-check set has finite scores, no categorical predicate triggers, and those checks exhaust the event decision. On such a stratum, the minimum field G satisfies Eq. (11): a negative score implies a confirmed failure and a positive score excludes one. At zero, the infimum need not be attained, so the original monitor determines the event and its time. Appendix A.2 proves these sign relations and the field's continuity under the stated response and context conditions.

Three masks separate input availability, supervision, and numerical validity. The evidence mask M records which inputs exist and their precision. The target-support mask V^{mon} supervises dynamic fields through the right-censored time, tied-mode quantities through the tie window, and continuous scoring margins on the fixed grid. Deterministic validity V^{num} marks physical or numerical definitions on that grid. Neither V^{mon} nor V^{num} is learned, and a predicted early event cannot shorten its own target.

For a required coordinate, $V^{\mathrm{num}} = 0$ exactly when the value is nonfinite, its identity or presence mask is inconsistent, its orientation cannot be normalized, or the simulator or restoration call fails. Unsupported coordinates are handled by V^{mon} instead. A required monitor coordinate with $V^{\mathrm{num}} = 0$ makes the raw response `INVALID`, whereas an extreme but finite physical value remains valid and is evaluated by the fixed monitor. The raw record Ψ contains named physical fields, derived checks, and V^{num} on the full grid; V^{mon} selects the task-supported part of Φ. When valid, Φ contains the dynamic path through first confirmation, tied-mode quantities through the tie window, full-horizon continuous scoring-margin traces, and event, time, and mode outputs. The aggregate scores g_f^{sc} and G are recomputed from $(\mu_t, \Psi; \kappa^\star, \lambda_t)$ and are not obtained by cropping Φ. The full post-failure trajectory is not a task target, and an invalid raw record yields the separate status `INVALID`. Reports keep valid no-failure, confirmed failure, `INVALID`, and `unavailable` distinct. All persistent checks use $\Delta_f = 0.25 t_g$, and the tied-mode window is $\epsilon_{\mathrm{tie}} = 0.05 t_g$.

B.4 Simulation, observation, and restoration

All public-suite targets use MuJoCo 3.13.0 with the `implicitfast` integrator, the Newton solver with 100 iterations and tolerance 10^{-10}, elliptic friction cones, and six-dimensional contacts. Contact compliance uses `solref` $(0.02 t_g, 1)$ and `solimp` $(0.95, 0.99, 0.001)$. The primary visual condition assigns density $250\,\mathrm{kg\,m^{-3}}$ and analytic inertia from each declared primitive and centered CoM, with sliding friction 0.45 between objects and 0.60 against the carrier, zero restitution, and torsional and rolling friction 0.005 and 0.0001 in both contact classes. These are global protocol conditions rather than hidden episode inputs.

An imported scene first undergoes a pre-anchor settling screen for at most $4t_g$; the query monitor is not yet running. The final snapshot becomes a resting reset only after all objects maintain linear speed below $0.01 L_{\mathrm{ref}}/t_g$ and angular speed below $0.02/t_g$ for $0.5 t_g$ and all reset-time checks are nonviolating. Failure to settle yields an unavailable anchor, not a no-failure label.

Three calibrated 384×384 red–green–blue (RGB) cameras use carrier-relative azimuth/elevation pairs $(-45^\circ, 30^\circ)$, $(45^\circ, 30^\circ)$, and $(180^\circ, 55^\circ)$; each is $2.5 L_{\mathrm{ref}}$ from and aimed at the carrier center, with vertical field of view 45° and near/far planes $0.01 L_{\mathrm{ref}}$ and $10 L_{\mathrm{ref}}$. A resting anchor supplies one synchronized three-view snapshot. A moving anchor supplies the full reset-to-anchor clip: 16 synchronized three-view samples, including both endpoints, over the preceding t_g, with no evidence after the query time. Primary renders use a neutral background and one fixed light rig.

Each fork manifest records the fields needed for exact restart: carrier state and clock, randomized physical parameters, query-anchor and support references, exact monitor memory μ_b, model and version identifiers, and candidate seed. At an admitted cut, restoration copies the manifest, executes only the selected legal continuation, and returns the shared physical target $\Psi(\chi_b, \lambda_b; q)$ for both routes (MuJoCo Developers, 2026). Before data generation, the audit independently restores a saved prefix, reruns its continuation, and compares named physical fields, predicates, event, time, and mode. A branch passes only if maximum position, orientation-geodesic, linear-velocity, and angular-velocity errors are at most $10^{-6} L_{\mathrm{ref}}$, 10^{-6} radians, $10^{-6} L_{\mathrm{ref}}/t_g$, and $10^{-6}/t_g$, respectively. Sampled predicates and tied modes must match exactly; confirmation times may differ by at most one physics step. Failed audits invalidate the branch, record the cause, and exclude it from training and evaluation. All executed simulator branches count against the budget, including invalid ones.

C Experimental Protocols and Additional Results

We detail the experiments in Section 6. Intervals are 95% CIs; counts distinguish independent parents or episodes from seed–query and seed–case evaluations.

C.1 Data and evaluation banks

PLP v0.0.3 contains approximately 32,000 layout/scenario entries, caps layouts at 20 objects, and corrects the Dataset 2 dimension mapping (Mazur, 2024; Mazur et al., 2027). We use its 10,560 base layouts as parents, keeping every loading prefix and CoM variant with its parent. A hash split with seed 2701 assigns parents to train/development/test in a 70/15/15 ratio, stratified by source subset and item-count bins $\{2{:}5, 6{:}10, 11{:}15, 16{:}20\}$. Resting candidates use 25%, 50%, 75%, and 100% of the loading order, rounded upward to at least two objects, with duplicate prefix lengths removed. The 19,434 ShapeStacks scenes retain the official parent split (Groth et al., 2018a;b). All derived views, programs, and forks inherit these splits. The two sources train separate models with the same carrier, cameras, materials, motion grammar, monitor, and simulator. Neither source's released stability labels nor PLP's ADAMS responses serve as our carrier-motion targets.

Table 4: Public-suite data accounting. Anchor and query counts describe the test split before model-seed replication; restoration failures are included in unavailable anchors.

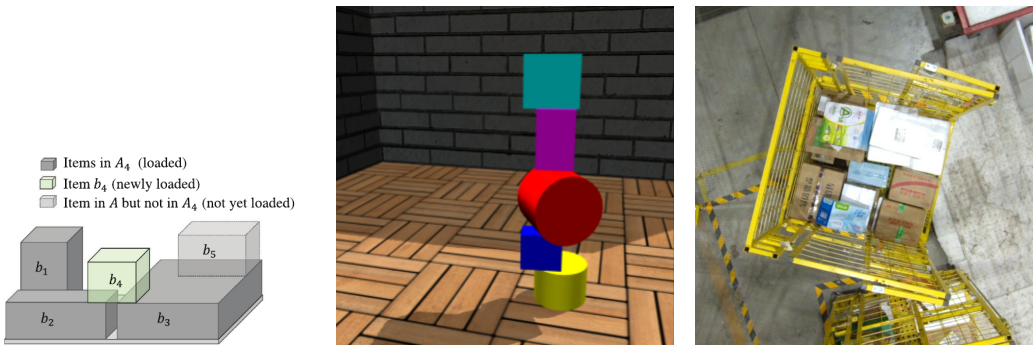
Unit	PLP	ShapeStacks
Parents: train / development / test	7,368 / 1,598 / 1,594	13,614 / 2,910 / 2,910
Scored test parents	1,589	1,443
Resting anchors: candidate / admitted	5,517 / 4,831	2,910 / 1,443
Resting anchors unavailable	686	1,467
Moving anchors: candidate / admitted	4,831 / 3,502	1,443 / 1,145
Moving anchors unavailable	1,329	298
Restoration failures	27	4
Response queries per model seed	1,066,624	331,264
Radius cases per model seed	256	256

Observation anchors. Resting anchors pass the settling, reset, and restoration checks in Appendix B.4. For each admitted resting anchor, a counter-based stream keyed by source, parent, anchor ID, and seed 2901 fixes a cut in $\{1, 1.5, 2\}t_g$ and orders training-family prefixes before their outcomes are observed. The first prefix legal through that cut supplies one moving candidate; a non-live, nonfinite, or restoration-invalid candidate is unavailable and is not replaced using a future outcome. This prefix is the physical history used to form the moving case. The moving query itself resets at the first of the 16 synchronized three-view samples in the preceding t_g clip: that registered reset fixes its support graph, pose references, and zero incoming dwell. Both clip endpoints are included, and no later image is available. The separate continuation study advances from one observation anchor by $\Delta/t_g \in \{0.5, 1, 1.5, 2\}$ and predicts another $2t_g$, with no cut observation.

Response and radius banks. The public suites use three 384×384 RGB views, horizon $4t_g$, and the motion grammar and output grid in Appendix B. Each response case receives 128 held-out-family programs: 64 chirps and 64 simultaneous three-axis compositions, accepted alternately from the stream keyed by $(\text{source}, \text{parent}, \text{anchor}, 3501)$. Illegal proposals and duplicates within $d_{\text{act,H}} < 10^{-8}$ are rejected in order; a bank still underfilled after 2,560 proposals is unavailable. For radius evaluation, the first legal held-out-family proposal from the corresponding seed-3701 stream is the reference and bank entry one, fixed before model or simulator outcomes are viewed. The other entries alternate between 512 local motions generated by Sobol-sampled changes to the free spline coefficients ($0 < d_{\text{act,H}} \leq 0.25$) and 511 proposals from the full legal coefficient domain. The same legality and duplicate checks apply, with a 20,480-proposal cap and no outcome-dependent replacement. Banks at $K \in \{64, 128, 256, 512, 1024\}$ are nested prefixes; simulator and model evaluate the same candidates, each minimizing over its own valid subset.

Industrial episodes. One episode is one pallet-building run logged from the online system deployed on a robotic palletizing line at a large logistics company. Immediately after placement 10, the system predicts terminal collapse from only the causal images and operating records then available. We retain 1,562 episodes from 120 pallet or order-template groups, with 390 collapses and 1,172 noncollapses. The placement-10 cutoff excludes 83 short episodes, 37 without complete terminal assessments, and 24 with outcome-revealing evidence at or before the cutoff. Near-duplicate templates are grouped before labels are examined, and model selection stays within each outer training fold. The final collapse bit supplies only a terminal endpoint, not first-failure or motion-response labels.

Dataset examples. Figure 5 illustrates all three data sources.



(a) PLP (b) ShapeStacks (c) Industrial palletizing

Figure 5: Dataset examples: (a) a PLP loading-layout schematic (CC BY 4.0) (Mazur et al., 2027), (b) an official ShapeStacks tower (CC BY-NC-SA 4.0) (Groth et al., 2018a), and (c) a pre-outcome frame from the deployed industrial system.

C.2 Models, Training, and Acquisition

The comparisons cover explicit mechanics, visual classification, action-conditioned world modeling, and direct response prediction. The first two are task adaptations; DINO-WM provides another visual prediction baseline, while Direct and the StackProbe controls isolate the proposed learning choices.

Privileged mechanics. Our Ramos-style reference follows the physics-engine approach of Ramos et al. (2017) using PyBullet 3.2.7. It receives exact anchor poses, velocities, carrier state and clock, collision primitives, mass, analytic inertia, contact coefficients, monitor memory, and reset support references. Bullet uses gravity g, step $0.0025t_g$, deterministic overlapping pairs, cone friction, and 100 sequential-impulse iterations; contact coefficients match the primary material condition. The kinematic carrier follows the same pre-integrated pose and velocity program at every step. Poses and velocities are resampled to the common grid, contact impulses become step-averaged forces, and the same carrier-frame support and monitor fields are recomputed. This baseline has no split route, since resetting it with exact state at the cut would introduce an additional oracle.

Visual baselines. The ShapeStacks adaptation trains the published Inception-v4 trunk from scratch on our failure-by-H labels (Groth et al., 2018a;b). Shared-trunk features are averaged over the three views and, for moving anchors, over the 16 frames. Declared context, registration, and the 3×6 spline coefficients enter a two-layer width-256 GELU MLP and are concatenated with those features. Development binary cross-entropy selects the checkpoint, with earlier-update ties; probability 0.5 defines event decisions and candidate-wise radius search. The classifier has no physical response, split state, time, or mode output.

DINO-WM retains a frozen DINOv2 patch encoder and an action-conditioned latent predictor (Zhou et al., 2025). Calibrated-view embeddings and two self-attention layers fuse the three grids; two causal temporal Transformer layers fuse moving-anchor frames. Acceleration integrated over $0.25t_g$ intervals is tokenized with the spline coefficients and interval endpoints. Autoregressive prediction targets frozen features from future renders at those intervals through $4t_g$. The physical reader uses

predicted latents, within-interval action, and declared context to decode the $0.025t_g$ grid. A causal-record decoder initializes monitor memory; split prediction advances the latent record and model-side memory, without a cut observation. This adaptation uses direct targets and coverage trajectories, without verified dual-route loss or failure-aware acquisition. The classifier, DINO-WM, Direct, and StackProbe + coverage share the attempted coverage ledger and use the labels their objectives support.

Architecture and optimization. StackProbe and its controls initialize from V-JEPA 2.1 checkpoint `vjepa2_1_vit_large_384` (Mur-Labadia et al., 2026; Meta FAIR, 2026). The three view streams share weights and use a four-layer width-512 fusion Transformer with eight heads, MLP ratio four, and dropout 0.1. The fused code has $d_z = 512$, the core has $d_c = 256$, and the predictor has six width-512 Transformer layers. The reader is a three-layer width-512 GELU MLP over object–time queries; the c_{512} control changes only core width under the same parameter-matching rule. Direct shares the observation encoder, fusion, response reader, monitor, and optimization, but maps the anchor and complete program to a response without a resumable state. Before training, predictor width is chosen from $\{384, 416, 448, 480, 512, 544, 576\}$ to match trainable parameters outside the shared trunk within 5%; a row with no qualifying width is unavailable.

Learned rows use five seeds, 100,000 updates, batch size 32, clipping at 1.0, 5% linear warmup, and cosine decay. AdamW uses $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon = 10^{-8}$, weight decay 0.05, and learning rates 10^{-4} for new modules and 10^{-5} for a trainable pretrained trunk. Runs use bfloat16 on four H100 80GB GPUs with FP32 loss reductions. Response checkpoints are evaluated every 5,000 updates by development direct nRMSE; ties within 10^{-4} use lower direct false-safe, then earlier update. Split, radius, and diagnostic banks do not enter model selection. Within StackProbe, separately normalized response, causal-record, and context losses have equal weight. Each verified continuation supervises both physical routes with total weight one; the stop-gradient EMA transition target has weight 0.05 and decay increasing from 0.996 to 0.9999.

Continuation controls. All five controls share a model-independent fork pool, restoration records, direct targets, paired seeds, reader, capacity rule, and training schedule. The pool has 16,000 initial attempted branches and six disjoint blocks of 8,000; invalid attempts remain charged and provide no physical target. One-future c uses every valid fork as a direct target, but only the first seed-ordered continuation per cut can supervise its split route, with no replacement if that continuation is invalid. Raw consistency replaces split physical supervision with normalized direct–split latent agreement. Advance z, verified c_{256}, and verified c_{512} use every verified direct/split target.

Acquisition comparison. The three selectors share the 16,000-branch seed replay, cuts, candidate pools, replay slots, initialization, optimizer schedule, and independent test banks. Their post-seed branches differ because branch choice is the intervention. Training reaches 25,000 updates at 16,000 calls, then 37,500, 50,000, 62,500, 75,000, 87,500, and 100,000 updates at the 24–64k checkpoints. Each attempted restoration or execution occupies a replay slot, including invalid attempts, without removal or replacement.

Each cut has 256 candidate motions: the reference, 127 local Sobol proposals within distance 0.25, and 128 broader legal proposals, using stream $(\text{source}, \text{parent}, \text{cut}, \text{round}, 4701)$. Common legality and duplicate checks precede scoring; pools underfilled after 5,120 proposals are excluded and counted. Coverage maximizes distance to the nearest previously attempted program. Margin disagreement allocates half of each post-seed block to coverage and half to the largest comparable direct–split monitor-margin difference. Failure-set acquisition retains that coverage share and otherwise chooses the nearest failure-bit disagreement, falling back to margin disagreement and then coverage. Ordering and ties are fixed before scoring, and selectors never inspect unexecuted simulator responses, events, target masks, or radii.

C.3 METRICS AND PUBLIC-SUITE RESULTS

For route $r \in \{D, S\}$, parent p, and query q, response nRMSE is

$$e^r_{p,q} = \left[\frac{1}{|\mathcal{G}_{p,q}|} \sum_{g \in \mathcal{G}_{p,q}} \frac{\sum_{j \in g} V^{\mathrm{mon}}_{p,q,j} d_g(\widehat{y}^r_{p,q,j}, y_{p,q,j})^2 / s_g^2}{\sum_{j \in g} V^{\mathrm{mon}}_{p,q,j}} \right]^{1/2}. \tag{25}$$

Groups weight position, linear velocity, angular velocity, orientation, force, and monitor margins equally, with scales L_{ref}, L_{ref}/t_g, $1/t_g$, one radian, object weight, and one. Distances are Euclidean, except for quaternion geodesic orientation and absolute margin error. The prediction-independent mask V^{mon} selects supported targets; groups with no supported coordinate are omitted, and queries with no supported group are invalid. Queries are averaged within parents, then equally across parents. W10 averages the largest $\lceil 0.1 N_{\mathrm{parent}} \rceil$ parent means, and bank maximum averages each parent's largest query error.

Direct prediction. Tables 5–7 expand the main results with CIs, response-query counts, and radius statuses. StackProbe lowers response error, false-safe, and false-alarm relative to Direct on both suites.

Table 5: Direct-route estimates and 95% CIs. FS and FA are percentages; mechanics is privileged, and the classifier's response is undefined.

Method	nRMSE	FS	FA	Radius cMAE
PLP				
Mechanics	.067 [.065, .068]	5.74 [5.59, 5.88]	4.99 [4.89, 5.09]	.014 [.010, .020]
ShapeStacks classifier	N/A	20.08 [19.16, 21.02]	15.94 [15.50, 16.42]	.036 [.031, .042]
DINO-WM + reader	.122 [.116, .129]	14.95 [14.21, 15.71]	12.47 [12.04, 12.89]	.026 [.023, .031]
Direct	.100 [.089, .110]	11.72 [10.44, 12.95]	10.90 [10.11, 11.66]	.020 [.017, .024]
StackProbe + coverage	.092 [.084, .101]	10.19 [9.31, 11.14]	9.10 [8.59, 9.64]	.017 [.014, .021]
StackProbe + acquisition	.083 [.078, .088]	7.80 [7.30, 8.33]	7.76 [7.43, 8.11]	.014 [.011, .016]
ShapeStacks				
Mechanics	.078 [.076, .079]	6.69 [6.48, 6.90]	5.94 [5.79, 6.09]	.014 [.010, .017]
ShapeStacks classifier	N/A	16.38 [15.52, 17.26]	13.36 [12.90, 13.83]	.036 [.031, .042]
DINO-WM + reader	.109 [.098, .122]	13.22 [11.84, 14.75]	11.19 [10.45, 11.98]	.023 [.019, .027]
Direct	.102 [.098, .106]	12.96 [12.43, 13.49]	11.81 [11.47, 12.18]	.019 [.016, .022]
StackProbe + coverage	.080 [.075, .085]	9.88 [9.23, 10.48]	9.04 [8.62, 9.43]	.015 [.012, .018]
StackProbe + acquisition	.078 [.071, .086]	8.79 [8.02, 9.68]	8.96 [8.43, 9.53]	.013 [.011, .015]

Among valid decisions, false-safe is $\mathrm{FN}/(\mathrm{TP} + \mathrm{FN})$ and false-alarm is $\mathrm{FP}/(\mathrm{TN} + \mathrm{FP})$. Time MAE conditions on joint confirmation; tied first modes use multilabel macro-F1, with an absent positive mode reported unavailable. Learned CIs use 10,000 paired draws over seeds and then a common resample of geometry parents across seeds and methods, retaining descendants together. Deterministic mechanics uses parent-cluster draws.

Table 6: Direct-route response-query counts. Failure and nonfailure columns contain valid simulator outcomes. Counts pool five learned seeds or one mechanics run.

Method	Numerical invalid	Failures	Nonfailures
PLP			
Mechanics	750	458,070	607,804
ShapeStacks classifier	693	2,291,784	3,040,643
DINO-WM + reader	15,211	2,285,248	3,032,661
Direct	6,406	2,289,313	3,037,401
StackProbe + coverage	8,888	2,288,215	3,036,017
StackProbe + acquisition	10,051	2,287,704	3,035,365
ShapeStacks			
Mechanics	224	143,271	187,769
ShapeStacks classifier	175	716,811	939,334
DINO-WM + reader	4,631	714,837	936,852
Direct	2,063	715,977	938,280
StackProbe + coverage	2,602	715,756	937,962
StackProbe + acquisition	3,082	715,530	937,708

Each radius case is classified as invalid reference, model/numerical invalidity, both found, simulator only, model only, or both not found. Conditional MAE uses both-found cases, including zero-radius failures; status mismatch is the one-sided-found count divided by all cases with valid references and model outputs. No-failure search outcomes remain censored, and zero true radius is evaluated by absolute error and reference-event agreement.

Table 7: Direct-route radius-case counts. NF means `not found`; counts pool five learned seeds or one mechanics run.

Method	Invalid reference	Model invalid	Both found	Simulator only	Model only	Both NF
PLP						
Mechanics	3	0	214	0	0	39
ShapeStacks classifier	15	0	1,057	13	31	164
DINO-WM + reader	15	4	1,063	4	13	181
Direct	15	8	1,058	4	14	181
StackProbe + coverage	15	6	1,063	2	9	185
StackProbe + acquisition	15	2	1,067	1	9	186
ShapeStacks						
Mechanics	1	1	227	0	0	27
ShapeStacks classifier	5	0	1,121	19	17	118
DINO-WM + reader	5	8	1,120	12	11	124
Direct	5	6	1,130	5	8	126
StackProbe + coverage	5	8	1,126	7	6	128
StackProbe + acquisition	5	8	1,130	3	6	128

Table 8 reports time and mode errors alongside confirmed, missed, and spurious events: the improved responses also yield more accurate predictions of when and how failure develops.

Table 8: First-confirmation time and tied-mode prediction. Time MAE is in t_g units and conditions on joint confirmation; intervals are 95% CIs.

	PLP		ShapeStacks	
Method	Time MAE ↓	Mode F1 ↑	Time MAE ↓	Mode F1 ↑
Direct	.171 [.153, .189]	.760 [.740, .782]	.175 [.168, .181]	.741 [.733, .749]
StackProbe	.135 [.127, .144]	.833 [.824, .841]	.125 [.114, .137]	.822 [.808, .835]

Source / method	Joint confirmations	Missed	Spurious
PLP / Direct	2,021,115	268,198	331,066
PLP / StackProbe	2,109,243	178,461	235,502
ShapeStacks / Direct	623,192	92,785	110,789
ShapeStacks / StackProbe	652,650	62,880	83,975

C.4 Continuation, acquisition, and radius analyses

Continuation supervision and the core. The five representation controls use 192 held-out PLP parents, five paired seeds, and 128 continuations in each of four prefix bins. Table 9 averages bins equally after query and parent aggregation; Figure 4a shows their prefix dependence. Supervising alternative continuations lowers split error and false-safe while retaining similar direct accuracy.

Table 9: Matched continuation controls (95% CIs). Direct and split errors are each measured against the simulator; FS is split-route false-safe in percent.

Control	Direct nRMSE	Split nRMSE	Split FS
One-future c	.087 [.083, .092]	.162 [.151, .173]	19.09 [18.21, 19.99]
Raw consistency	.087 [.082, .091]	.162 [.152, .171]	19.11 [18.34, 19.88]
Advance z	.084 [.080, .089]	.127 [.119, .136]	16.36 [15.55, 17.20]
Verified c_{256}	.084 [.080, .088]	.103 [.097, .109]	14.03 [13.35, 14.71]
Verified c_{512}	.082 [.077, .086]	.098 [.091, .106]	13.53 [12.78, 14.33]

Table 2 reports the paired c_{256} contrasts. The c_{512} control improves over Advance z by .029 [.019, .037]. Both tested core widths improve continuation prediction, supporting a dedicated representation for state propagation.

Information use. Table 10 changes one cut input at evaluation. The nulling and parent-shuffling interventions show that the evolved core carries information used by the split predictor for response and failure prediction.

Table 10: Evaluation-only information-path interventions, with 95% CIs. True memory/context are privileged; all rows retain a .21% [.19, .23] numerical-invalid rate.

Cut input	Split nRMSE	Split false-safe (%)
Predicted core, memory, context	.103 [.097, .109]	14.03 [13.35, 14.71]
True monitor memory	.103 [.097, .109]	13.62 [12.88, 14.34]
True declared context	.099 [.093, .105]	13.65 [12.95, 14.33]
Null core	.266 [.250, .283]	52.74 [52.32, 53.17]
Parent-shuffled core	.219 [.205, .233]	55.38 [54.96, 55.79]

Acquisition. Curve area integrates split false-safe at $\{16, 24, 32, 40, 48, 56, 64\} \times 10^3$ attempted calls over a horizontal span normalized to $[0, 1]$. Table 11 reports this area and the final checkpoint, with paired coverage contrasts in Table 2. Prioritizing failure-bit disagreement yields the lowest curve area and endpoint among the three selectors, making the fixed simulation budget more effective for failure detection.

Table 11: PLP acquisition and full-suite response-error tails (95% CIs). FS area and endpoint are percentages; valid/invalid training slots pool five runs (320,000 attempts per rule).

Rule	FS area	FS at 64k	Valid / invalid slots
Coverage	15.28 [13.92, 16.60]	12.57 [11.54, 13.57]	318,788 / 1,212
Margin disagreement	14.03 [12.70, 15.42]	10.90 [9.71, 12.19]	318,600 / 1,400
Failure-set disagreement	12.48 [11.39, 13.69]	8.32 [7.50, 9.27]	318,545 / 1,455

Split-response nRMSE on the full PLP suite

Rule	W10	Bank maximum
Coverage	.206 [.187, .226]	.473 [.433, .514]
Failure-set disagreement	.208 [.184, .230]	.533 [.475, .590]

Finite-bank radius. Table 12 uses nested banks for the direct-route acquisition model. Each bank has 1,280 seed–case evaluations, of which 15 simulator references and two model references are invalid. Conditional cMAE decreases from .018 to .014 as banks grow, measuring agreement on the distance from a plan to a tested failure. The table tracks the changing both-found cohort and one-sided findings at each bank size.

Table 12: Nested-bank radius accounting on PLP. The first panel excludes the 17 invalid references; NF means `not found`. The second counts invalid nonreference candidates and reports the minimum across cases of the largest tested action distance.

K	cMAE [95% CI]	Both found	Simulator only	Model only	Both NF
64	.018 [.015, .022]	1,064	4	4	191
256	.016 [.013, .020]	1,066	2	7	188
1024	.014 [.011, .016]	1,067	1	9	186

K	Model invalid / calls	Simulator invalid	Minimum reach
64	122 / 79,569	40	.391484
256	467 / 322,065	215	.404699
1024	1,904 / 1,292,049	945	.405640

Appendix A.5 examines the local radius mechanism separately from these candidate-bank results.

C.5 INDUSTRIAL TERMINAL-OUTCOME TRANSFER

We evaluate the deployment logs at the placement-10 cutoff, using five outer folds to produce out-of-fold predictions for every retained episode. Each fold uses a group-disjoint inner split for head selection, calibration, and the operating threshold. The four inputs are records alone, records plus frozen V-JEPA 2.1 features, records plus frozen Direct without continuation, and records plus frozen PLP-trained StackProbe c (Mur-Labadia et al., 2026; Meta FAIR, 2026). Frozen branches receive the same images available by the cutoff, camera calibration, and carrier registration; identity, geometry, mass, placement time, and realized pose enter once through the record stream. StackProbe's public-suite context is masked unavailable rather than filled from those records. This comparison isolates the core's image-conditioned feature against generic frozen video features.

At placements 1–10, a linear projection combines a width-128 record embedding with the frozen feature, or a fixed zero vector for records alone. The ten width-256 tokens enter a two-layer, four-head causal Transformer and a two-layer width-256 GELU–LayerNorm head with dropout 0.1. Five matched head seeds use class-balanced binary cross-entropy and AdamW with learning rate 10^{-3}, $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$, weight decay 10^{-4}, batch size 64, and at most 200 epochs. Inner-development Brier selects early stopping with patience 20; temperature scaling minimizes development NLL. The threshold maximizes sensitivity subject to development FPR at most 10%, over unique scores and an all-negative sentinel, breaking ties by higher threshold then fixed index.

Each seed pools all five out-of-fold partitions before computing average precision (AUPRC); CIs resample template groups and matched seeds. Table 13 reports ranking and operating-point behavior. The AUPRC gain over records alone shows that the frozen simulation-trained core provides useful information for collapse prediction in real palletizing; Section 6 reports the paired contrast.

Table 13: Placement-10 predictions on industrial deployment logs, evaluated out of fold with 95% CIs. The threshold maximizes sensitivity subject to a development FPR of at most 10%.

Input	AUPRC ↑	False-negative (%)	False-positive (%)
Records	.558 [.507, .614]	54.46 [49.72, 59.27]	10.19 [8.61, 11.86]
Records + frozen V-JEPA	.644 [.592, .696]	45.49 [40.51, 50.53]	9.86 [8.36, 11.45]
Records + frozen Direct	.672 [.621, .721]	41.18 [36.21, 46.29]	9.78 [8.24, 11.38]
Records + frozen StackProbe	.732 [.684, .778]	32.82 [28.21, 37.76]	9.73 [8.19, 11.34]