# ATOMIC PHYSICAL TRANSITIONS: A STRUCTURED REPRESENTATION FOR PHYSICAL DYNAMICS IN VIDEO

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Current video understanding often compresses multiple physical processes into clip-level labels or answers, allowing correct event-level predictions without explicitly recovering the underlying state changes. These processes can instead be represented as compositions of reusable causal units—physical state transitions defined by the mechanisms that drive them rather than by scene appearance. We introduce **Atomic Physical Transitions (APTs)**: minimal, temporally localized state changes that bind visual evidence to an active mechanism and a before/after physical state, forming an ordered transition chain for each video. For diagnosis and supervision, we construct **APT-BENCH**, covering 14 transition types across contact, gravity, friction, and rotation/stability, with approximately 55,000 timed instances over 2,500 trials. A human-anchored, rule-guided simulation pipeline scales annotation using realized state traces rather than intended scene templates. Evaluation of twelve vision-language models reveals low zero-shot transition recovery, and a denser-input check shows that the recovery gap persists. APT-only fine-tuning improves detection but can degrade event-level answering. We therefore introduce **APT-Tune**, a parameter-efficient recipe combining multimodal answer-only masking with format-conditional co-training across APT, multiple-choice, and description targets. Across six open-source backbones, it substantially improves transition recovery while retaining event-level performance on MVBench. APT-Tune consistently improves performance on PhysBench across the evaluated backbones. These results support APTs as an intermediate representation between coarse event recognition and full physical simulation, useful for both diagnosis and transferable supervision. The project page is at https://apt-physical-transitions-2027.netlify.app/

## 1 INTRODUCTION

A ball bouncing on a table is more than a *bounce* label: it loses support, falls, makes contact, rebounds, and eventually settles. These state changes reveal how the event unfolds. Recognizing the event name alone does not specify when contact begins, when motion reverses, or when the object separates from the surface. Mechanics organizes such changes through gravity, contact, collision, friction, and stability (Ling et al., 2016). Physics experts emphasize underlying principles over surface features (Chi et al., 1981; Larkin et al., 1980), while event cognition studies meaningful boundaries in continuous activity (Zacks & Swallow, 2007). Together, these observations motivate locating the transitions within a physical event rather than describing only its overall outcome.

Existing physics-oriented video benchmarks evaluate interactions, prediction, counterfactual reasoning, and physical plausibility (Yi et al., 2019; Bear et al., 2021; Tung et al., 2023; Chow et al., 2025; Motamed et al., 2025; Tragoudaras et al., 2025; Mak et al., 2026). Their clip-level answers and event labels often leave intermediate state changes implicit. A correct answer about an outcome does not by itself establish whether a model has identified the contacts, support changes, and motion transitions within the event. We argue for an ordered chain of timed transitions, each associating a visible state change with a mechanism-typed label. Such chains form an intermediate representation between event recognition and full physical simulation: they expose selected physical changes without requiring reconstruction of every underlying physical variable.

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.

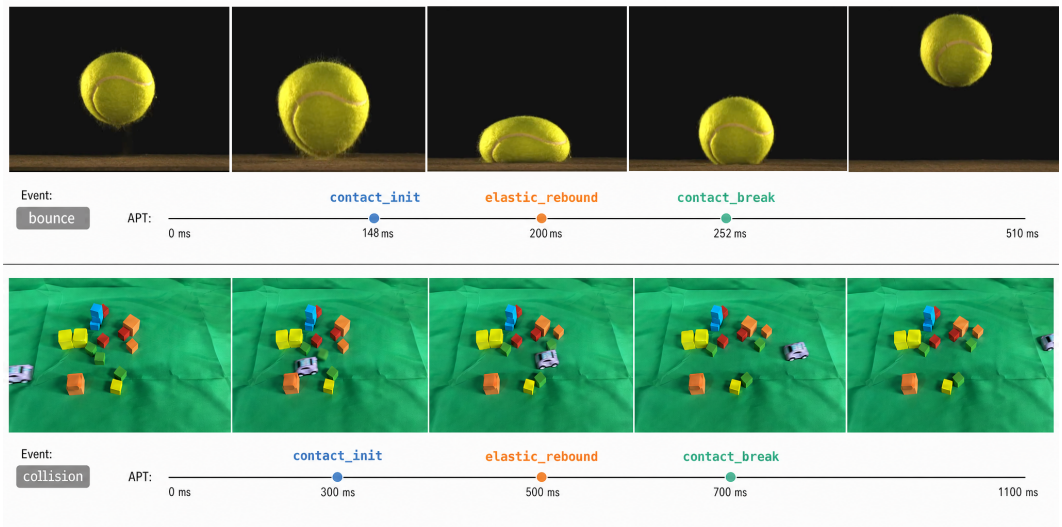


Figure 1: **Event labels summarize outcomes; APT chains localize physical transitions.** A clip label such as *bounce* or *collision* summarizes an event. Atomic Physical Transitions (APTs) mark typed changes and times, such as `contact_init`, `elastic_rebound`, and `contact_break`. The same types apply to visually different contact scenes within our macroscopic-mechanics taxonomy.

To operationalize this view, we introduce **Atomic Physical Transitions** (APTs): visually localizable changes between coarse physical states under an active mechanism. An APT associates a time and type with a before/after state change; a video becomes an ordered *APT chain*. Our taxonomy covers 14 types in macroscopic mechanics, defined by physical processes rather than object appearance. Here, atomicity is relative to the chosen vocabulary and observable video timescale, rather than implying physically indivisible processes. APT-BENCH scores types and timestamps, without independently verifying mechanism explanations or causal links (Section 3.2).

APT-level supervision is hard to obtain by annotation alone: clips may contain several brief transitions whose boundaries must be identified consistently. We build a mixed-source, human-calibrated pipeline. Five annotators label controlled CLEVRER and Physion++ videos (Yi et al., 2019; Tung et al., 2023), producing 27,303 APTs in 1,246 trials. Rule-guided simulation adds 1,254 trials and approximately 27,700 APTs, for 2,500 trials and approximately 55,000 APTs in total. These added labels follow realized contact, motion, and support changes in simulator traces rather than intended scene templates, with human validation of visual recognizability.

Using this data, we test whether vision-language models (VLMs) can recover APT chains. On the 500-trial test set (10,200 APTs), twelve zero-shot VLMs achieve only 10.0–15.2% recall under the shared 32-frame, 14-type protocol; missed transitions dominate the error. On Qwen3-VL-2B, correctly masked APT-only fine-tuning raises recall to 33.8%, while SSv2 accuracy decreases from 72.5 to 68.5%. Improved transition recovery therefore does not automatically preserve event-level answering. This trade-off motivates combining transition supervision with rehearsal across ordinary video answer formats.

We address this with `APT-Tune`, a parameter-efficient SFT recipe combining APT, multiple-choice, and description records. APT chains provide localized transition targets, while SSv2 multiple-choice and description records provide event-level rehearsal through format-conditional co-training. Multimodal answer-only masking excludes image-padding tokens from the loss, and LoRA adapts language-model projections while the vision encoder remains frozen. On Qwen3-VL-2B, the full recipe raises APT recall from 10.0 to 38.1% and F1 from 14.8 to 38.8%; SSv2 accuracy rises from 72.5 to 87.5%, and PhysBench accuracy from 38.5 to 50.8% without PhysBench training data. Across six backbones, transition recall improves and event-level benchmark averages are retained or improved. These gains characterize the combined training recipe, including event-level rehearsal.

Our contributions are threefold:

- **We propose APTs as a human-inspired physical transition representation.** APTs represent videos as ordered chains of visually grounded, mechanism-typed state changes, making within-event transitions and their temporal locations explicit.
- **We construct mixed-source APT data for diagnosis and supervision.** Human annotation and rule-guided simulation provide approximately 55,000 timed APTs across 2,500 trials. APT-BENCH uses these annotations to evaluate transition recovery and reveal substantial gaps under a shared evaluation protocol.
- **We introduce `APT-Tune` for learning with transition-level supervision.** The recipe combines APT-chain targets with event-level rehearsal, improving transition recovery and PhysBench performance across six backbones while retaining or improving average MVBench accuracy.

## 2 RELATED WORK

Physics-oriented benchmarks primarily score answers, outcomes, plausibility, or trajectories (Yi et al., 2019; Bear et al., 2021; Tung et al., 2023; Chow et al., 2025; Motamed et al., 2025). APT complements them by making localized, mechanism-typed state changes explicit targets for annotation, evaluation, and supervision. Drawing on event segmentation and principle-based physics expertise (Chi et al., 1981; Larkin et al., 1980), APT models physical regime transitions rather than semantic actions or procedural steps in temporal video understanding (Goyal et al., 2017; Damen et al., 2018; Lea et al., 2016; Farha & Gall, 2019). `APT-Tune` combines this transition-level causal supervision with format-conditional co-training and parameter-efficient adaptation (Liu et al., 2023; Lin et al., 2023; Hu et al., 2021) for transfer across QA and description formats. See Appendix A for a full discussion.

Table 1: **Comparison of evaluation units across physics and temporal video benchmarks.** APT differs from prior benchmarks by making timestamped, mechanism-typed physical transitions the supervision and scoring unit, enabling ordered chain-level evaluation.

| | Evaluation unit | | Physical / temporal structure | | | Provenance | |
|---|---|---|---|---|---|---|---|
| **Data / Eval.** | **Target** | **Granularity** | **Temporal Location** | **Physics Taxonomy** | **Ordered Chain** | **Label / Source** | **Scale (native)** |
| CLEVRER (2019) | Causal video QA | Event QA | ✗ | Collision events | Partial | Sim GT | 20K vids / >300K QA |
| CLEVRER-Humans (2023) | Causal event QA | Event / edge QA | ✗ | Free-form events | Graph | Human on sim | 1,108 vids / 1,076 QA |
| Physion (2021) | Physical prediction | Future outcome | Endpoint | Object dynamics | ✗ | Sim GT + human | 1.2K examples / 8 scen. |
| Physion++ (2023) | Physical prediction | Future outcome | Endpoint | Latent properties | ✗ | Sim GT + human | 8K train / 768 test[‡] |
| PhysBench (2025) | Physical VLM QA | Entry QA | ✗ | Broad domains | ✗ | Unspecified[†] | 10,002 entries |
| Physics-IQ (2025) | Video continuation | Future continuation | Partial | Physical laws | ✗ | Real video GT | 396 vids / 66 scen. |
| MASS-Bench (2025) | Physics video QA | Entity motion | Partial | Physics categories | Partial | Human+AIGC | 4,350 vids / 8,361 QA |
| TemporalBench (2024) | Temporal QA / caption | Sub-event QA | Partial | Temporal only | Partial | Human + LLM | 10K QA / 2K ann. |
| **APT (ours)** | **APT chain recovery** | **Transition chain** | **Timestamp** | **14 APTs / 4 domains** | ✓ | **Human + Sim GT** | **∼55K APTs / 2,500 trials** |

## 3 ATOMIC PHYSICAL TRANSITIONS: FROM EVENT LABELS TO MECHANISM TYPED CHAINS

### 3.1 FROM EVENT OUTCOMES TO PHYSICAL STATE CHANGES

A clip-level event label names what happened, but collapses the causal process that makes it happen: a "bounce" may include contact initiation, collision response, contact break, and settling. This matches how mechanics is learned: reusable physical processes such as gravity, contact, collision, friction, rotation, and stability are learned separately, while real events compose them over time. We therefore define Atomic Physical Transitions (APTs) by *physical process* rather than object or material category. A transition on a deformable or fluid-like object can share a represented mechanism with a rigid-body transition when the same local state change is visible; this does not extend the taxonomy to general non-rigid dynamics. Event labels name outcomes; APT chains make the constituent physical changes explicit. Detailed motivation is given in Appendix B.1.

### 3.2 DEFINITION

**Definition.** Given a video $V$ of a physical scene, an **Atomic Physical Transition** is a visually localizable moment at which a physical entity, material region, or interacting pair changes between




Figure 2: **Simulator-derived APT examples.** Rendered frames illustrate the 14 types across varied scenes; labels follow local temporal state changes and realized simulation traces.

two coarse physical states under a specific active mechanism. We represent an APT as

$$a_i = (\tau_i, d_i, \ell_i, s_i^- \rightarrow s_i^+, e_i), \tag{1}$$

where $\tau_i$ is the canonical transition time, $d_i$ is the physical domain, $\ell_i \in \mathcal{L}_{d_i}$ is the transition type, $s_i^- \rightarrow s_i^+$ is the before/after physical state change, and $e_i$ is the local visual evidence supporting the transition. The domain $d_i$ groups transitions by active physical process rather than by object class or material type; the type $\ell_i$ names the specific state change within that process family; and the evidence $e_i$ anchors the label to local visual cues. Thus material affects how an APT appears, but not what the APT means: an APT records when a state changes and associates that change with a mechanism in the taxonomy.

Atomicity is relative to the 14-type vocabulary and the video timescale: an APT marks the smallest named, visually localizable change for an entity or interacting pair. The annotation rule favors physically meaningful boundaries over arbitrary subdivisions; for example, `sliding_arrest` and later `settling` describe distinct changes. Appendix B.6 gives the granularity rules.

**What scoring $(\tau_i, \ell_i)$ measures.** The representation in Eq. (1) associates a mechanism, directed state change, and evidence with each transition, but the benchmark scores only type and timestamp. Each type encodes a canonical mechanism–state-change pair: for example, `free_fall_onset` denotes supported→falling dynamics, and `elastic_rebound` denotes approach→retreat after contact. Correctly localizing these labels measures recovery of a mechanism-typed transition representation. It does not independently verify the mechanism, evidence attribution, entity identity, or causal links between successive transitions. Physical coupling rules guide data construction, but the matching metric scores typed timestamps rather than those rules between transitions. Intervention, counterfactual reasoning, and explicit mechanism explanation require separate evaluation.

### 3.3 Problem Formulation: APT Chain Enumeration

Let $V = (I_1, \ldots, I_N)$ be a video with $N$ native frames and duration $T$. Transition-level physical understanding requires recovering an ordered chain of APTs:

$$\mathbf{A}(V) = (a_1, \ldots, a_K), \quad 0 \le \tau_1 \le \cdots \le \tau_K \le T, \qquad a_i^{\text{eval}} = (\tau_i, \ell_i), \tag{2}$$

where $K$ is unknown and varies by video, and each evaluation item is serialized as a typed timestamp. The task is therefore to recover how many transitions occur, when they occur, and which physical type

each transition has. Timestamps locate each transition in a common coordinate across the different durations and frame rates of our sources (Table 3); order or sampled-frame indices alone cannot express this consistently. The model must predict an unknown number of transition instances, their types, and their times. Appendix B.4 discusses alternative serializations and mechanism-conditioned decoding.

### 3.4 APT TAXONOMY

Our current taxonomy focuses on visually grounded macroscopic mechanics, primarily rigid-body and force-mediated transitions involving contact, collision, gravity, friction, rotation, and stability. A represented transition may also occur on a deformable or fluid-like object when its local state change is governed by one of these mechanisms, but the taxonomy does not cover general fluid or deformable-body dynamics. APTs are process-centered: labels are determined by the active mechanism and visible before/after state change, not by object or material category. They cover localizable transitions such as contact onset/loss, support loss, collision response, friction- or medium-mediated motion change, rotation onset/arrest, stability loss, and settling, while excluding invisible changes, continuous parameter estimation, or full physical-state reconstruction. Labels are assigned to the entity, region, or interacting pair undergoing the transition. Each instance receives one label: annotators select the most specific type that explains its visible state change. Distinct changes can share a timestamp, such as `contact_init` and `landing` at an impact, because they describe contact onset and the onset of support respectively. Appendix B.6 gives the full disambiguation rules. Controlled clips contain several interacting objects, while dense contact cascades and severe occlusion remain under-represented (Appendix E.1).

Table 2: **APT taxonomy.** APTs are organized by physical process domain and defined by coarse before/after state changes.

| **Domain I: Contact & Collision** | | | **Domain IV: Rotation & Stability** | | |
|---|---|---|---|---|---|
| **Label** | **Before** | **After** | **Label** | **Before** | **After** |
| `contact_init` | separated | interacting | `rotation_onset` | non-rotating | rotating |
| `contact_break` | interacting | separated | `toppling` | stable | unstable |
| `elastic_rebound` | approach | retreat | `rotation_arrest` | rotating | stationary |
| `inelastic_capture` | moving | captured/rest | `settling` | disturbed | equilibrium |
| **Domain II: Gravity & Projectile** | | | **Domain III: Friction, Surface & Medium** | | |
| **Label** | **Before** | **After** | **Label** | **Before** | **After** |
| `free_fall_onset` | supported | falling | `static_breakaway` | stationary | moving |
| `projectile_apex` | ascending | descending | `sliding_arrest` | moving | stationary |
| `landing` | falling | supported | `surface_transition` | surface A | surface B |

Our taxonomy contains 14 recurring transitions grouped by active physical process. The same represented transition can occur on different materials when its mechanism and visible before/after state change match; our evaluation centers on controlled, mechanics-oriented scenes. **Contact & collision** captures interaction-boundary changes and outcomes; **gravity & projectile** captures support loss, apex reversal, and landing; **friction, surface & medium** captures rest-to-motion, motion-to-rest, and surface/medium changes; and **rotation & stability** captures angular-motion changes, toppling, and final equilibrium.

## 4 APT DATA CONSTRUCTION

Human annotations on controlled CLEVRER and Physion++ videos define the 14 APT types, yielding 27,303 timed APTs over 1,246 trials. Five annotators follow shared guidelines; each video is independently labeled by two annotators for transition type and timestamp, with disagreements adjudicated by a third. Pre-adjudication labels were not retained, so inter-annotator $\kappa$ is unavailable.

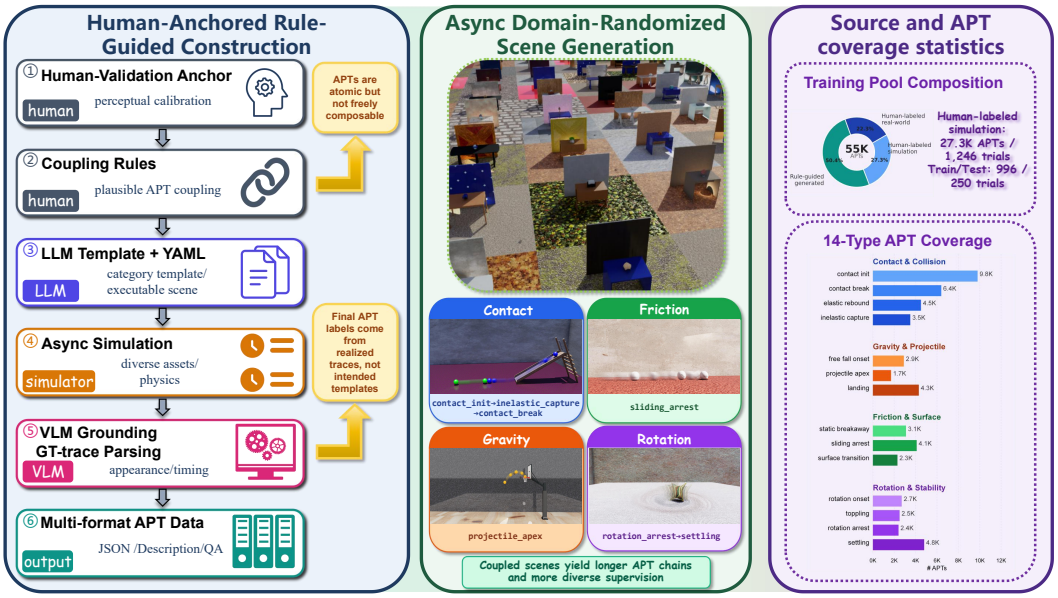


Figure 3: **APT data construction.** Human annotation anchors transition semantics; rule-guided scene generation and simulator traces supply scalable examples, with labels determined by realized dynamics. The right panel summarizes source composition and type coverage.

Rule-guided simulation adds 1,254 trials and ∼27,700 APTs, producing a dataset of 2,500 trials and ∼55,000 APTs, split into 2,000 training and 500 test trials by source and scenario.

To scale beyond the human-labeled anchor, we use the rule-guided pipeline in Figure 3. Human-defined transition couplings are expanded by an LLM into scene templates and converted by a coding agent into executable YAML. Domain-randomized simulation renders videos and records state traces; a VLM grounds visible objects, and an LLM-assisted parser converts the traces into timed APT chains. Final labels follow realized contact, motion, and support changes rather than intended templates, with human validation ensuring visual consistency. Appendix C provides construction details.

Figure 2 samples simulated clips across all 14 APT types. Repeated labels show different objects and materials; each type refers to a local before/after change observed over the clip, not the displayed frame alone. The human-source test transitions are human labeled; simulator test transitions use rule-derived labels checked for visual validity.

**Evaluation input.** All default APT-BENCH comparisons use 32 uniformly sampled frames. Each frame is preceded by an explicit millisecond timestamp, and the prompt also provides clip duration and native frame rate. Predictions are matched one-to-one to test APTs by type within $\pm 200$ ms. Table 3 summarizes source durations and nominal sampling density; the complete prompt, parsing rules, and matching protocol appear in Appendix C.12.

Table 3: **APT-BENCH test-split statistics by source.** The nominal 32-frame interval is duration divided by 32, rounded to milliseconds; actual sampled timestamps depend on native-frame selection. Source duration and frame-rate differences motivate millisecond-valued timestamps and the frame-budget analysis in Appendix D.5.

| Source | #Test | Duration (s) | FPS | Resolution | 32-frame interval (nominal) | APTs/clip |
|---|---|---|---|---|---|---|
| CLEVRER | 100 | 5.12 | 25 | 480×320 | ≈ 160 ms | 12.0 |
| Physion++ | 150 | 12.89 (7.00–21.87) | 30 | 256×256 | ≈ 403 ms | 29.0 |
| Sim Data | 250 | 7.13 (5.00–10.00) | 30 | 480×320 | ≈ 223 ms | 18.6 |
| **All** | **500** | **8.46** | — | — | **≈ 264 ms** | **20.4** |

## 5 APT-TUNE: LEARNING FROM APT CHAINS

APT-Tune is a learning realization of the APT representation. Training only on APT chains can favor a structured-output specialist, so we combine APT enumeration with multiple-choice and

description records to retain prompt-dependent video answering. The recipe has two components: multimodal answer-only masking and format-conditional co-training.

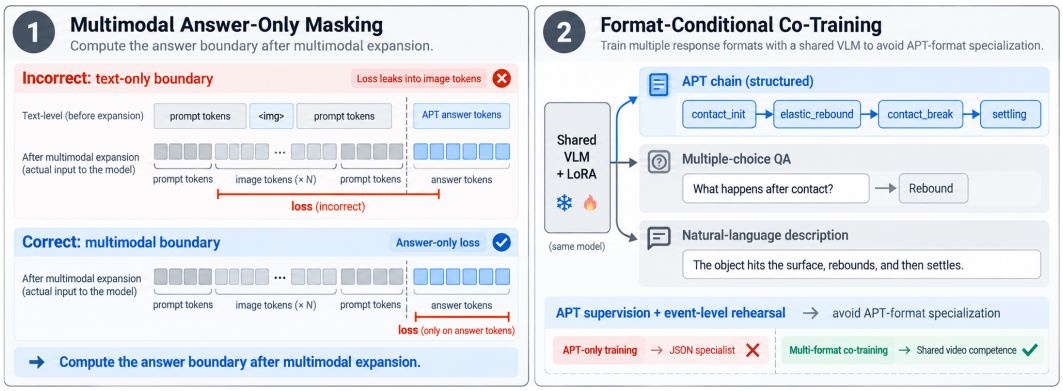


Figure 4: **`APT-Tune` training recipe.** Answer-only multimodal masking supervises APT output tokens, while APT, multiple-choice, and description records share one backbone to retain prompt-dependent answer formats.

The multimodal processor expands image placeholders into visual tokens. Computing the supervised prefix length from text alone can mistakenly include those visual tokens in the loss, so we determine the answer boundary after multimodal processing. The format mix uses one APT-chain example from each of 1,000 APT video trials alongside 1,000 SSv2 (Goyal et al., 2017) multiple-choice and 1,000 short-description records, for 3,000 training examples in total. We freeze the vision encoder and apply LoRA (Hu et al., 2021) to language-model attention and MLP projections. Appendix C.13 details the masking and training mix; Appendix C.12 specifies the frame-time input.

## 6 Experiments

We evaluate zero-shot APT recovery, `APT-Tune` across six backbones, transfer beyond APT enumeration, and the roles of answer masking and format mixing. Unless noted, APT-Bench results use 500 test trials (10,200 APTs), 32 timestamped frames, a shared 14-type prompt, and one-to-one type-and-time matching within $\pm 200$ ms. The evaluation requires recovering individual transition instances and their temporal locations rather than predicting a single clip-level label. Protocols and further results appear in Appendices C.12 and D.

### 6.1 Zero-Shot Diagnostic

**Metrics.** Recall measures the fraction of annotated transitions recovered with both the correct type and an in-tolerance timestamp. Precision, F1, and prediction counts complement recall by characterizing unmatched outputs and output volume. Auxiliary Type is type-only recall from one-to-one matching by label without a time constraint (Appendix C.12), measuring type recovery independently of temporal matching. Median timing error $\Delta T$ is computed only over type-and-time matched predictions and therefore does not characterize missed transitions.

**Findings.** Twelve systems achieve only 10.0–15.2% zero-shot recall (Table 4). Qwen3-VL rises from 10.0 to 12.0% between 2B and 8B, whereas InternVL3.5 stays near 13–14%. Increasing model size therefore yields limited or inconsistent gains within these families under the shared protocol. Type-only recall ranges from 12.0 to 25.6%, indicating that removing the timing constraint does not eliminate the recovery gap. Matched-event median $\Delta T$ is 67–110 ms, but applies to the small subset of successfully recovered transitions. Under this protocol, missed transitions dominate the error.

### 6.2 `APT-Tune` Across Six Open-Source Backbones

**Recipe.** Qwen3-VL and InternVL3.5 at 2B, 4B, and 8B share a 3,000-example SFT mix: one APT chain from each of 1,000 APT video trials, plus 1,000 SSv2 multiple-choice and 1,000 description records. The SSv2 records provide event-level rehearsal alongside the structured transition targets.

Table 4: **Zero-shot APT-BENCH results, 32 frames.** Recall and time-agnostic type-only recall (Type) in %; $\Delta T$ is median matched-event timing error (ms). Temporal matching tolerance: $\pm 200$ ms.

| Model | Rec. | Type | $\Delta T$ | Model | Rec. | Type | $\Delta T$ |
|---|---|---|---|---|---|---|---|
| Gemini 2.5 Flash | 12.0 | 14.0 | 110 | Qwen3-VL-2B | 10.0 | 12.0 | 100 |
| GPT-5 | 13.6 | 22.0 | 67 | Qwen3-VL-4B | 11.0 | 13.0 | 95 |
| Gemini 3.1 | 12.8 | 23.1 | 67 | Qwen3-VL-8B | 12.0 | 14.0 | 90 |
| Claude Sonnet 5 | **15.2** | **25.6** | 67 | InternVL3.5-2B | 14.0 | 16.0 | 67 |
| VideoLLaMA-3-2B | 12.4 | 16.5 | 95 | InternVL3.5-4B | 13.0 | 15.0 | 75 |
| VideoLLaMA-3-7B | 13.9 | 18.4 | 80 | InternVL3.5-8B | 13.0 | 15.0 | 80 |

LoRA adapts language-model projections with the vision encoder frozen; the answer-only masking rule and three task formats are shared, using each backbone's native chat template. This setup evaluates the same training recipe across two model families and three parameter scales. Implementation details appear in Appendix C.13.

**Results.** `APT-Tune` improves recall by 28.1–40.4 percentage points across all six backbones (Table 5); the lowest tuned recall (38.1%) exceeds the best zero-shot result (15.2%). Type-only recall rises from 12.0–16.0% to 52.1–71.2%, and median $\Delta T$ on each model's matched events falls from 67–100 to 19–38 ms. From 2B to 8B, tuned recall increases by 10.5 points for Qwen3-VL and 6.6 points for InternVL3.5. These size-related gains are more pronounced than in the zero-shot setting, although substantial transition-recovery errors remain even after tuning.

Table 5: **`APT-Tune` on six backbones.** Recall and time-agnostic type-only recall (Type) in %; matched-event $\Delta T$ in ms; $\Delta$Rec is recall gain (pp) at $\pm 200$ ms. Best per column in bold.

| | Zero-shot | | | `+APT-Tune` | | | |
|---|---|---|---|---|---|---|---|
| **Backbone** | **Rec.** | **Type** | $\Delta T$ | **Rec.** | **Type** | $\Delta T$ | **$\Delta$Rec.** |
| Qwen3-VL-2B | 10.0 | 12.0 | 100 | 38.1 | 52.1 | 38 | 28.1 |
| Qwen3-VL-4B | 11.0 | 13.0 | 95 | 42.5 | 57.8 | 33 | 31.5 |
| Qwen3-VL-8B | 12.0 | 14.0 | 90 | 48.6 | 66.2 | 27 | 36.6 |
| InternVL3.5-2B | **14.0** | **16.0** | **67** | 46.8 | 61.5 | 25 | 32.8 |
| InternVL3.5-4B | 13.0 | 15.0 | 75 | 48.0 | 63.5 | 23 | 35.0 |
| InternVL3.5-8B | 13.0 | 15.0 | 80 | **53.4** | **71.2** | **19** | **40.4** |

**Prediction-count control.** To examine whether recall gains arise only from producing more predictions, we analyze output counts on Qwen3-VL-2B (Table 6). Predictions per clip rise from 7.1 to 19.6 (ground truth: 20.4), while precision increases from 28.8 to 39.6% and F1 from 14.8 to 38.8%. Capping tuned outputs at each clip's zero-shot prediction count before matching still yields 28.2% recall, compared with 10.0% for the zero-shot model. This count-controlled gain indicates that increased output volume alone does not explain the improvement. All 14 types improve in both precision and recall (Appendix D.3).

Table 6: **Prediction-count control on Qwen3-VL-2B.** All columns use one-to-one type-and-time matching within $\pm 200$ ms on the 500-trial test set (10,200 ground-truth APTs, 20.4 per clip). The last column caps `APT-Tune` at the zero-shot model's per-clip prediction count before matching.

| Model | #Pred (per clip) | Prec. | Rec. | F1 | Rec. @ capped count |
|---|---|---|---|---|---|
| Qwen3-VL-2B (zero-shot) | $\sim$3,542 (7.1) | 28.8 | 10.0 | 14.8 | 10.0 |
| + APT-Tune | $\sim$9,814 (19.6) | **39.6** | **38.1** | **38.8** | **28.2** |

### 6.3 TRANSFER TO EVENT-LEVEL VIDEO AND OOD PHYSICS

We evaluate transfer beyond APT enumeration using a 32-frame budget on held-out MVBench (Li et al., 2023) and PhysBench (Chow et al., 2025). MVBench tests whether transition-focused adaptation preserves event-level video understanding, while PhysBench evaluates performance on physical questions outside the training data. MVBench gains reflect the full recipe, including SSv2 rehearsal; no PhysBench data are used for training.

**MVBench (event-level).** All six backbones improve their average accuracy across the 20 tasks by 0.5–2.2 percentage points (Table 12). The gains occur in both model families and at all three model sizes, indicating that improved transition recovery does not come at the expense of aggregate performance on this event-level benchmark. We interpret these modest gains as evidence of performance retention under the full training recipe, rather than uniform improvement on every individual task.

**PhysBench (OOD physics).** The official average improves by 11.1–14.2 percentage points across the six backbones, with gains on all four reported subtasks (Table 7). For example, Qwen3-VL-2B improves from 38.5% to 50.8%, while InternVL3.5-8B rises from 45.2% to 59.4%. Dynamics shows the largest gain for every backbone (+16.6 to +20.3 points), whereas Scene gains the least in five of six cases. These results show that the benefits of the full recipe extend beyond structured APT outputs to physical question answering. However, the broad subtask categories combine multiple phenomena and do not identify which specific physical processes drive the improvements.

Table 7: **PhysBench transfer (%).** Official Avg and four subtask scores; bold marks gains within each backbone.

| Backbone | Variant | Avg | Property | Relationships | Scene | Dynamics |
|---|---|---|---|---|---|---|
| Qwen3-VL-2B | Zero-shot | 38.5 | 46.9 | 38.5 | 29.8 | 38.2 |
| | +APT-Tune | **50.8** | **58.5** | **49.5** | **36.5** | **56.7** |
| Qwen3-VL-4B | Zero-shot | 41.9 | 50.2 | 44.1 | 31.4 | 40.1 |
| | +APT-Tune | **53.2** | **61.9** | **54.5** | **38.2** | **57.3** |
| Qwen3-VL-8B | Zero-shot | 45.0 | 55.5 | 50.1 | 31.8 | 42.1 |
| | +APT-Tune | **56.6** | **64.5** | **59.9** | **39.2** | **59.5** |
| InternVL3.5-2B | Zero-shot | 40.2 | 50.5 | 40.2 | 30.1 | 39.1 |
| | +APT-Tune | **51.3** | **61.2** | **50.5** | **35.8** | **55.7** |
| InternVL3.5-4B | Zero-shot | 43.7 | 52.5 | 46.2 | 32.1 | 42.3 |
| | +APT-Tune | **55.2** | **65.8** | **56.8** | **39.0** | **60.7** |
| InternVL3.5-8B | Zero-shot | 45.2 | 57.1 | 50.3 | 30.5 | 42.4 |
| | +APT-Tune | **59.4** | **68.5** | **62.2** | **43.5** | **62.7** |

**Mechanism-selective transfer.** To examine this pattern more closely, we compare two selected groups of PhysBench subtypes on Qwen3-VL-8B. Five motion/spatial subtypes gain 18.4 points on average, while five non-mechanics subtypes change by $-0.1$ points (Appendix D.2, Table 13). This contrast is consistent with the mechanics-oriented scope of the APT taxonomy. However, Viewpoint measures spatial interpretation rather than a physical mechanism, and the two groups are not an exhaustive partition of PhysBench. We therefore treat this analysis as a descriptive diagnostic for one backbone, not a general claim of mechanism-selective transfer across all models.

**Training recipe and frame budget.** On Qwen3-VL-2B, correctly masked APT-only SFT raises APT recall from 10.0 to 33.8% and PhysBench accuracy from 38.5 to 44.7%; adding SSv2 rehearsal further improves these scores to 38.1% and 50.8%, respectively (Table 15). SSv2-only SFT reaches 11.6% APT recall and 40.2% PhysBench accuracy. This comparison changes both the training data and supervision targets, so it evaluates the full recipe rather than isolating the contribution of APT structure. The masking ablation separately checks correct implementation of multimodal answer-only supervision. With 64 input frames, tuned recall remains higher than zero-shot recall (39.7% versus 18.8%), showing that the tuning advantage persists beyond the default frame budget (Appendix D.5).

## 7 Conclusion

APT represents physical events as ordered chains of localized, mechanism-typed state changes, providing an intermediate representation between coarse event labels and full physical simulation. APT-Bench reveals that current VLMs recover these transition chains poorly in the zero-shot setting, while APT-Tune consistently improves transition recovery across six open-source backbones and remains effective under denser temporal input. Transfer on PhysBench further shows that the gains concentrate in mechanics-aligned subtypes, consistent with the scope of the current APT taxonomy. These results support APT as both a diagnostic representation and a useful supervision target for physical video understanding. Future work should extend APT evaluation to denser multi-object interactions, stronger occlusion, and unconstrained real-world video.

## ETHICS STATEMENT

This work introduces a transition-level supervision unit for physical video understanding, with applications in robotics, embodied agents, scientific video analysis, and safety-oriented evaluation of multimodal models, where APT chains can expose brittle physical reasoning that clip-level QA hides. The benchmark uses short physical transitions in synthetic and controlled real videos rather than personal, medical, or otherwise sensitive data, so direct misuse risk is low. Indirectly, improved physical video understanding could feed into automation or surveillance systems; we mitigate this by releasing only the dataset, evaluation protocols, and training recipes, with no high-risk generative model and no scraped personal data.

## LLM USAGE DISCLOSURE

We used large language models (LLMs) as supportive tools for manuscript preparation and implementation. Specifically, LLMs assisted with language polishing, including improving clarity, grammar, and conciseness; literature discovery to help identify potentially missing related work; and partial experimental implementation, including code writing, debugging, organization, and execution support through Claude Code. All LLM-assisted code was reviewed, modified, tested, and validated by the authors before use. All substantive scientific ideas, methods, experiments, analyses, results, and conclusions are original contributions of the authors. The authors reviewed and verified all LLM-assisted content and take full responsibility for the manuscript.

## REFERENCES

Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Neural Information Processing Systems*, 2016. URL https://api.semanticscholar.org/CorpusID:2200675.

Daniel M. Bear, Elias Wang, Damian Mrowca, Felix J. Binder, Hsiau-Yu Fish Tung, R. T. Pramod, Cameron Holdaway, Sirui Tao, Kevin A. Smith, Fan-Yun Sun, Fei-Fei Li, Nancy G. Kanwisher, Joshua B. Tenenbaum, Daniel Yamins, and Judith E. Fan. Physion: Evaluating physical prediction from vision in humans and machines. *ArXiv*, abs/2106.08261, 2021. URL https://api.semanticscholar.org/CorpusID:235436143.

Mu Cai, Reuben Tan, Jianrui Zhang, Bocheng Zou, Kai Zhang, Feng Yao, Fangrui Zhu, Jing Gu, Yiwu Zhong, Yuzhang Shang, Yao Dou, Jaden Park, Jianfeng Gao, Yong Jae Lee, and Jianwei Yang. Temporalbench: Benchmarking fine-grained temporal understanding for multimodal video models. *ArXiv*, abs/2410.10818, 2024. URL https://api.semanticscholar.org/CorpusID:273346169.

Michelene T. H. Chi, Paul J. Feltovich, and Robert Glaser. Categorization and representation of physics problems by experts and novices. *Cogn. Sci.*, 5:121–152, 1981. URL https://api.semanticscholar.org/CorpusID:44436293.

Wei Chow, Jiageng Mao, Boyi Li, Daniel Seita, Vitor Campagnolo Guizilini, and Yue Wang. Physbench: Benchmarking and enhancing vision-language models for physical world understanding. *ArXiv*, abs/2501.16411, 2025. URL https://api.semanticscholar.org/CorpusID:275932005.

Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The epic-kitchens dataset. *ArXiv*, abs/1804.02748, 2018. URL https://api.semanticscholar.org/CorpusID:4710439.

Yazan Abu Farha and Juergen Gall. Ms-tcn: Multi-stage temporal convolutional network for action segmentation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3570–3579, 2019. URL https://api.semanticscholar.org/CorpusID:67877065.

T. Fetaya, Elias Wang, K.-C. Welling, Michelle Zemel, Thomas Kipf, Ethan Fetaya, Kuan-Chieh Jackson Wang, Max Welling, and Richard S. Zemel. Neural relational inference for interacting systems. *ArXiv*, abs/1802.04687, 2018. URL https://api.semanticscholar.org/CorpusID:215716912.

Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter N. Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. The "something something" video database for learning and evaluating visual common sense. *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 5843–5851, 2017. URL https://api.semanticscholar.org/CorpusID:834612.

Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 961–970, 2015. URL https://api.semanticscholar.org/CorpusID:1710722.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *ArXiv*, abs/2106.09685, 2021. URL https://api.semanticscholar.org/CorpusID:235458009.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1988–1997, 2017. URL https://api.semanticscholar.org/CorpusID:15458100.

Jill H. Larkin, John P. McDermott, Dorothea P. Simon, and Herbert A. Simon. Expert and novice performance in solving physics problems. *Science*, 208:1335 – 1342, 1980. URL https://api.semanticscholar.org/CorpusID:10639995.

Colin S. Lea, Michael D. Flynn, René Vidal, Austin Reiter, and Gregory Hager. Temporal convolutional networks for action segmentation and detection. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1003–1012, 2016. URL https://api.semanticscholar.org/CorpusID:8738977.

Kunchang Li, Ya-Li Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Lou, Limin Wang, and Yu Qiao. Mvbench: A comprehensive multi-modal video understanding benchmark. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22195–22206, 2023. URL https://api.semanticscholar.org/CorpusID:265466214.

Bin Lin, Yang Ye, Bin Zhu, Jia-Xi Cui, Munan Ning, Peng Jin, and Li Yuan. Video-llava: Learning united visual representation by alignment before projection. *ArXiv*, abs/2311.10122, 2023. URL https://api.semanticscholar.org/CorpusID:265281544.

Samuel J. Ling, Jeff Sanny, and William Moebs. *University Physics Volume 1*. OpenStax, Rice University, 2016. URL https://api.semanticscholar.org/CorpusID:126216483.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *ArXiv*, abs/2304.08485, 2023. URL https://api.semanticscholar.org/CorpusID:258179774.

Chak-Wing Mak, Guanyu Zhu, Boyi Zhang, Hongji Li, Xiaowei Chi, Kevin Zhang, Yichen Wu, Yangfan He, Chunkai Fan, Wentao Lu, Kuangzhi Ge, Xinyu Fang, Hong-Yang He, Kuan Lu, Tianxiang Xu, Li Zhang, Yong-Xin Ni, Youhua Li, and Shanghang Zhang. Physicsmind: Sim and real mechanics benchmarking for physical reasoning and prediction in foundational vlms and world models. *ArXiv*, abs/2601.16007, 2026. URL https://api.semanticscholar.org/CorpusID:284935206.

Jia-Yuan Mao, Xuelin Yang, Xikun Zhang, Noah D. Goodman, and Jiajun Wu. Clevrer-humans: Describing physical and causal events the human way. *ArXiv*, abs/2310.03635, 2023. URL https://api.semanticscholar.org/CorpusID:255188817.

Saman Motamed, Laura Culp, Kevin Swersky, Priyank Jaini, and Robert Geirhos. Do generative video models understand physical principles? *2026 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 948–958, 2025. URL https://api.semanticscholar.org/CorpusID:275570164.

Nvidia. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *ArXiv*, abs/2511.04831, 2025. URL https://api.semanticscholar.org/CorpusID:282889617.

Antonios Tragoudaras, Chenyu Zhang, Daniil Cherniavskii, Antonios Vozikis, Thijmen Nijdam, Derck W. E. Prinzhorn, Mark Bodracska, Niculae Sebe, Andrii Zadaianchuk, and Efstratios Gavves. Evaluating newtonian mechanics in video generative models with real physical systems. 2025. URL https://api.semanticscholar.org/CorpusID:277596368.

Hsiao-Yu Fish Tung, Mingyu Ding, Zhenfang Chen, Daniel M. Bear, Chuang Gan, Joshua B. Tenenbaum, Daniel L. K. Yamins, Judy E. Fan, and Kevin A. Smith. Physion++: Evaluating physical scene understanding that requires online inference of different physical properties. *ArXiv*, abs/2306.15668, 2023. URL https://api.semanticscholar.org/CorpusID:259261994.

Xiyang Wu, Zongxia Li, Jihui Jin, Guangyao Shi, KV Gouthaman, Vishnu Raj, Nilotpal Sinha, Jingxi Chen, Fan Du, and Dinesh Manocha. Mass: Motion-aware spatial-temporal grounding for physics reasoning and comprehension in vision-language models. *ArXiv*, abs/2511.18373, 2025. URL https://api.semanticscholar.org/CorpusID:283243727.

Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B. Tenenbaum. Clevrer: Collision events for video representation and reasoning. *ArXiv*, abs/1910.01442, 2019. URL https://api.semanticscholar.org/CorpusID:203641884.

Jeffrey M. Zacks and Khena M. Swallow. Event segmentation. *Current Directions in Psychological Science*, 16:80 – 84, 2007. URL https://api.semanticscholar.org/CorpusID:32043826.

# A Extended Related Work

## A.1 Physics-Oriented Video and Multimodal Benchmarks

A large body of recent work evaluates physical reasoning from visual input. CLEVRER studies descriptive, explanatory, predictive, and counterfactual reasoning over collision-centric synthetic videos (Yi et al., 2019). Physion evaluates whether humans and models can predict the physical outcome of simulated dynamic scenes, while Physion++ further stresses online inference of latent physical properties such as mass, friction, elasticity, and deformability (Bear et al., 2021; Tung et al., 2023). PhysBench broadens the evaluation scope to VLMs and interleaved video-image-text physical reasoning (Chow et al., 2025). More recent work evaluates whether generated or mixed-modality videos obey physical principles, including Physics-IQ, Morpheus, MASS/MASS-Bench, and PhysicsMind (Motamed et al., 2025; Tragoudaras et al., 2025; Wu et al., 2025; Mak et al., 2026). These benchmarks are highly relevant, but they primarily score answers, endpoint predictions, plausibility decisions, generated trajectories, or grounded QA targets. Our APT framework is organized around a different unit: the ordered sequence of temporally localized, mechanism-typed transitions inside an event. APTBench is the diagnostic instantiation of this framework, while APT-Tune uses the same transition representation as training supervision.

## A.2 Event Cognition and Physics Expertise

APTs are motivated by two classic observations from cognitive science. First, event segmentation research argues that observers parse continuous activity into meaningful temporal units, and that these boundaries support memory, prediction, and learning (Zacks & Swallow, 2007). Second, studies of physics expertise show that experts tend to organize problems by underlying physical principles rather than surface-level objects or scene appearance (Chi et al., 1981; Larkin et al., 1980). APTs combine these intuitions in a video setting: they are boundary-based, but the boundary is defined by a change in active physical mechanism. Thus, an APT is not merely a shorter event label; it specifies a before/after physical state, a visible cue, and the mechanism that makes the transition occur.

## A.3 Temporal Video Understanding and Action Segmentation

Large-scale video datasets and temporal segmentation methods have established that temporal structure is central to video understanding. Something-Something emphasizes fine-grained object interactions and visual commonsense, ActivityNet supports activity classification and temporal localization, and EPIC-KITCHENS provides dense egocentric action and object annotations (Goyal et al., 2017; Heilbron et al., 2015; Damen et al., 2018). Temporal action segmentation further assigns frame- or segment-level semantic action labels over long videos (Lea et al., 2016; Farha & Gall, 2019). Our framework is adjacent to this literature, but changes the unit of segmentation. Instead of semantic actions such as pushing, pouring, or picking up, APTs mark physical regime changes such as support loss, contact initiation, rebound, sliding arrest, and settling. These transitions may occur within a single action label, or a single transition type may recur across many visually different actions.

## A.4 Object-Centric Dynamics and Simulation

Object-centric dynamics models represent physical scenes through entities, relations, and interactions. Interaction Networks and Neural Relational Inference are representative examples: they use structured relational representations to learn or infer physical dynamics (Battaglia et al., 2016; Fetaya et al., 2018). Synthetic physical reasoning benchmarks such as CLEVRER, Physion, and Physion++ similarly benefit from simulator access and controlled physical state information (Yi et al., 2019; Bear et al., 2021; Tung et al., 2023). APTs are complementary to these approaches. Rather than replacing full state rollout, object-centric prediction, or simulation, APTs provide a compact transition representation that can be read by humans, evaluated as a transition chain, and used as supervision for VLMs. This makes APTs a bridge between low-level simulator traces and high-level event-level QA.

### A.5 Causality, Counterfactuals, and Explanation

Our work is also related to diagnostic visual reasoning and causal explanation. CLEVR introduced a controlled benchmark for compositional visual reasoning, and CLEVRER extended this diagnostic style to temporal and causal video reasoning (Johnson et al., 2017; Yi et al., 2019). Physics-oriented benchmarks further test whether models can reason about counterfactual outcomes, physical plausibility, and law-consistent prediction or generation (Chow et al., 2025; Motamed et al., 2025; Tragoudaras et al., 2025; Mak et al., 2026). The distinction is that answer-level causal evaluation can still leave the intermediate process latent. A model may correctly identify the responsible object or final outcome without enumerating the mechanism changes that connect the initial and final states. APT chains make these intermediate transitions first-class labels and metrics.

### A.6 VLM Instruction Tuning and Structured-Output Fine-Tuning

Recent multimodal instruction-tuning pipelines show that visual and video models can be improved by supervised fine-tuning on mixed conversational, QA, and caption-style data (Liu et al., 2023; Lin et al., 2023). Parameter-efficient methods such as LoRA make such adaptation practical for large models (Hu et al., 2021). However, transition supervision introduces a different risk: if the model is trained only to emit structured APT strings, it may learn a schema rather than a transferable representation. APT-Tune is designed to avoid this failure mode. It uses APT enumeration as a transition-level pretext task, while format-conditional co-training preserves ordinary event-level answering across multiple-choice, QA, and short-description prompts.

### A.7 Dataset Construction and Annotation

Our data construction is related to both human annotation for temporally structured video and simulator-derived supervision. ActivityNet and EPIC-KITCHENS illustrate how human annotations can support temporal localization and fine-grained video understanding (Heilbron et al., 2015; Damen et al., 2018). CLEVRER, Physion, and Physion++ illustrate the value of controlled simulation and privileged physical state information for systematic physical reasoning evaluation (Yi et al., 2019; Bear et al., 2021; Tung et al., 2023). The APT framework combines these strengths. Human labels anchor the semantics of transition types, while simulator ground truth provides contacts, poses, velocities, and support states that can be translated into the same APT schema. The resulting labels are not merely event names or endpoint outcomes; they are typed physical transitions that support both evaluation through APTBench and representation learning through APT-Tune.

## B Supplementary Details for Atomic Physical Transitions

This appendix provides additional details for the Atomic Physical Transition (APT) formulation introduced in Sec. 3. The main paper gives a compact definition and taxonomy. Here we expand the motivation, annotation scope, boundary rules, and label-level cues of the APT representation. As clarified in Section 3.2, benchmark scoring measures typed transition localization; mechanism explanations, causal edges, and interventions are not independently evaluated.

### B.1 Extended Motivation: From Event Names to Causal Processes

A clip-level event label is an outcome description. It tells the reader what the whole video can be called, but not which physical transitions make that outcome valid. For example, the event label "bounce" may refer to a clip in which an object approaches a surface, initiates contact, undergoes collision response, reverses direction, breaks contact, and later lands or settles. The label compresses these steps into one name. An APT chain recovers the hidden causal process by marking the short-lived physical state changes that compose the event.

This distinction mirrors how physical events are usually explained. Introductory mechanics teaches reusable mechanisms such as gravity, contact, collision, friction, rotation, and stability as separate topics. Real scenes then compose these mechanisms over time. A falling-and-bouncing object is not understood by the word "bounce" alone, but by the sequence of support loss, gravity-driven motion, contact initiation, collision response, separation, and eventual damping or settling. Thus APTs are

not intended to be finer event names. They are mechanism-typed transition units: each APT binds a local visual cue to an active physical mechanism and a before/after state change.

This also separates three objects that are often conflated. The *physical event* is the full clip-level outcome, such as falling, bouncing, colliding, or toppling. The *APT chain* is the structured transition representation that explains the outcome. The *output format* is only a serialization of that representation, such as JSON, multiple-choice answering, QA, or free-form description. This distinction is central to our learning setup: if a model treats APT supervision only as a JSON format, it can become an APT-format specialist; if it learns APTs as a shared transition representation, the same structure can support both local transition enumeration and ordinary event-level understanding.

### B.2 Expanded Formal Interpretation

In the main text, an APT is represented as

$$a_i = (\tau_i, d_i, \ell_i, s_i^- \to s_i^+, e_i), \tag{3}$$

where $\tau_i$ is the canonical transition time, $d_i$ is the physical domain, $\ell_i$ is the transition type, $s_i^- \to s_i^+$ is the before/after physical state change, and $e_i$ is the local visual evidence. This representation should be interpreted as follows.

**Canonical time.** The timestamp $\tau_i$ denotes the canonical visual moment of transition. For contact onset, this is the first frame in which a visible gap closes. For contact loss, it is the first frame with visible separation. For velocity reversal, it is the local moment where approach changes to retreat. For rest-to-motion or motion-to-rest transitions, it is the first visually stable frame at which the new regime becomes evident. When a transition unfolds over several frames, annotators choose the most characteristic point of the change rather than an arbitrary midpoint.

**Physical domain.** The domain $d_i$ identifies the mechanism family. It is not merely a coarse label group. It expresses which physical mechanism explains the transition: contact constraints and collision response, gravity and projectile motion, friction and surface interaction, or rotation and stability. This domain-level structure admits mechanism-conditioned decoding (Appendix B.12).

**Transition type.** The type $\ell_i$ identifies the named APT within the domain. For example, `contact_init` and `contact_break` describe the existence of a contact boundary, while `elastic_rebound` and `inelastic_capture` describe the physical outcome of the contact. Similarly, `free_fall_onset`, `projectile_apex`, and `landing` describe different phases of gravity-driven motion rather than a single coarse "falling" event.

**Before/after state.** The state transition $s_i^- \to s_i^+$ is intentionally coarse. We do not require complete physical state reconstruction, such as full 3D pose, mass, force magnitude, coefficient of friction, or coefficient of restitution. Instead, the before/after state captures the physical distinction that is visually meaningful and causally relevant: free versus contact, supported versus falling, sliding versus stationary, stable versus unstable, rotating versus non-rotating.

**Visual evidence.** The evidence $e_i$ anchors the label to the video. An APT should not be assigned solely because a hidden physical variable could have changed. The transition must have visible support, such as gap closure, visible separation, downward motion after support loss, direction reversal after impact, deceleration into rest, onset of spin, loss of balance, or disappearance of residual motion.

**Atomicity.** Atomicity is relative to our vocabulary and the visible timescale of the video. An APT is atomic if the same local state change does not contain another named APT that better explains it for the same object or object pair. This does not mean that the underlying physics is mathematically indivisible. For example, a collision involves continuous contact forces, deformation, and energy transfer, but the APT `elastic_rebound` marks the observable transition from approach to retreat. APTs are therefore the smallest mechanism-typed units that we ask models to localize, classify, and reuse across answer formats.

### B.3 Atomicity and Temporal Coordinates

Atomicity is defined relative to our vocabulary and video timescale. An APT is atomic if, for the same entity or interacting pair, the local neighborhood of $\tau_i$ contains no other named APT that further explains the same transition $s_i^- \to s_i^+$. In this sense, APTs are not physically indivisible events, but the smallest mechanism-typed units we ask models to localize, classify, and reuse across answer formats. Granularity is therefore set by physical meaning and visual observability rather than by subdividing the vocabulary arbitrarily: a transition is admitted only if it is physically meaningful, supported by a local neighborhood of at least three native video frames during annotation, and atomic under the current vocabulary. Thus `sliding_arrest` marks the visible end of translational sliding, while residual wobble that later disappears is labeled `settling` instead of being decomposed into individual oscillations; sub-frame deformation and force oscillations need a different capture timescale and lie outside the benchmark.

*Why timestamps rather than frame indices or order alone.* Sequence order records only which transition precedes which, whereas a timestamp additionally binds each transition to its local visual evidence, separates repeated instances of the same type, and gives a common coordinate across the differing frame rates and durations of our sources (Table 3); a frame index is equivalent only under a single fixed frame rate and sampling scheme. This is not a reduction to frame classification: the model must still predict the unknown number, types, instances, and order of transitions, and the contribution is the ordered mechanism-grounded chain rather than any particular temporal coordinate. An APT chain is a structured physical-transition representation rather than a fixed output style: the same chain can be serialized as JSON, verbalized as a description, or queried through QA prompts. This distinction matters because treating APT supervision as a JSON template can produce an APT-format specialist, whereas training across answer formats may help the model use transition supervision on the evaluated event tasks.

### B.4 APT Chains and Output Serializations

Given a video $V = (I_1, \ldots, I_T)$, an APT chain is an ordered sequence

$$\mathbf{A}(V) = (a_1, \ldots, a_K), \qquad 0 \le \tau_1 \le \cdots \le \tau_K \le T, \tag{4}$$

where $K$ varies by video. In evaluation, we serialize each APT as a typed timestamp:

$$a_i^{\text{eval}} = (\tau_i, \ell_i). \tag{5}$$

This serialization is convenient for matching predictions to ground truth, but it is not the representation itself.

For example, the same physical process can be represented in multiple answer formats:

```
APT enumeration:
[
  {"time": 0.84, "type": "contact_init"},
  {"time": 0.96, "type": "elastic_rebound"},
  {"time": 1.04, "type": "contact_break"},
  {"time": 1.72, "type": "landing"},
  {"time": 2.10, "type": "settling"}
]

Description:
"The ball hits the table, bounces away, lands again, and settles."

QA:
Q: What happens after the ball contacts the table?
A: It rebounds and then separates from the table.

Multiple choice:
The ball (A) passes through the table, (B) rebounds after contact,
(C) remains floating, or (D) disappears. Answer: B.
```

All four outputs can be grounded in the same APT chain. This motivates format-conditional co-training in the main method. APT supervision should improve the shared video representation, not collapse the model into one response style.

### B.5 SCOPE

Our current taxonomy focuses on visually grounded macroscopic mechanics, primarily rigid-body and force-mediated transitions involving contact, collision, gravity, friction, rotation, and stability. A represented transition may also occur on a deformable or fluid-like object when its local state change follows one of these mechanisms; this does not imply coverage of general fluid or deformable-body dynamics. We include physical state changes that satisfy three criteria.

**Physical meaningfulness.** The transition corresponds to a meaningful change in physical regime, such as contact onset, support loss, collision response, friction-mediated arrest, surface-dependent motion change, rotation onset, toppling, or final settling.

**Visual locality.** The transition has a visible cue in the video and can be localized to a frame or short temporal neighborhood. We do not label transitions that require hidden variables without visible evidence.

**Vocabulary-level atomicity.** The transition is not better explained by another named APT in the same local neighborhood for the same object or object pair.

We do not attempt to estimate continuous physical parameters such as mass, force magnitude, friction coefficient, restitution coefficient, full 3D pose, or exact contact forces. We exclude general fluid flow, fracture, thermal change, melting, tearing, and deformation mechanisms outside the represented local state changes. In this sense, APTs form an intermediate representation between clip-level event recognition and full physical simulation. They expose the order of mechanism-typed state changes without requiring complete state reconstruction.

### B.6 BOUNDARY AND DISAMBIGUATION RULES

APT labels are assigned to the object or object pair undergoing the transition. Multiple APTs may share the same timestamp if they refer to different objects or independent physical changes. For the same object or object pair, labels should not duplicate the same explanatory transition. When multiple labels could plausibly apply, annotators choose the label that best explains the active mechanism, preferring more specific transitions over generic ones.

**Generic versus specific transitions.** Some labels describe generic boundary changes, while others describe mechanism-specific outcomes. For example, `contact_init` marks the onset of contact, while `elastic_rebound` marks the collision response that reverses motion after contact. Both may occur in the same bounce sequence, but they mark different causal steps.

**Landing versus generic contact.** `landing` is used when an airborne or falling object becomes supported by a surface. `contact_init` is used for generic contact onset when support formation is not the primary transition, such as a rolling object touching a wall or two objects colliding laterally. If a falling object touches a surface and immediately rebounds, annotators may mark `contact_init` for the first contact and `elastic_rebound` for the reversal; if the object becomes supported, `landing` captures the falling-to-supported transition.

**Free fall onset versus generic contact break.** `free_fall_onset` is used when an object loses support and gravity becomes the dominant cause of subsequent motion. For example, a block sliding past the edge of a platform should be marked as `free_fall_onset`. `contact_break` is used for generic separation when the main consequence is loss of contact rather than the onset of gravity-driven falling, such as two horizontally interacting blocks separating.

**Elastic rebound versus contact break.** `elastic_rebound` marks the visible reversal from approach to retreat after collision. `contact_break` marks the later loss of contact, when the

Table 8: **Common APT disambiguation rules.** When two labels seem applicable to the same local change, annotators choose the label that best explains the active mechanism.

| Confusable labels | Decision cue | Guideline |
|---|---|---|
| `landing` vs. `contact_init` | support formation vs. generic contact | Use `landing` when a falling object becomes supported; use `contact_init` for lateral or generic contact onset. |
| `free_fall_onset` vs. `contact_break` | gravity-driven fall vs. generic separation | Use `free_fall_onset` when support loss causes falling; use `contact_break` when separation is the main transition. |
| `elastic_rebound` vs. `contact_break` | velocity reversal vs. contact loss | Use `elastic_rebound` for approach-to-retreat reversal; use `contact_break` for visible separation. |
| `inelastic_capture` vs. `sliding_arrest` | collision capture vs. frictional stop | Use `inelastic_capture` when impact dissipates motion into rest; use `sliding_arrest` when sliding motion decays to rest. |
| `sliding_arrest` vs. `settling` | end of sliding vs. final equilibrium | Use `sliding_arrest` for translational sliding-to-rest; use `settling` for final stabilization after residual motion. |
| `rotation_arrest` vs. `settling` | angular stop vs. full stabilization | Use `rotation_arrest` when visible angular motion stops; use `settling` when all residual motion ends. |
| `toppling` vs. `rotation_onset` | loss of stability vs. generic angular onset | Use `toppling` when an object loses stability and falls over; use `rotation_onset` for generic start of spin or angular motion. |
| `surface_transition` vs. `landing` | surface change vs. falling-to-supported | Use `surface_transition` when a supported object crosses surfaces; use `landing` when a falling object becomes supported. |

objects or the object and surface visibly separate. In a bounce, these may be close in time but they are not the same transition: rebound is the change in motion direction caused by collision response, while contact break is the change in contact state.

**Inelastic capture versus sliding arrest.** `inelastic_capture` is used when a moving object strikes another object or surface and is captured by that interaction, dissipating motion into rest at or against the contact. `sliding_arrest` is used when an object is already sliding along a surface and gradually comes to rest due to friction. The key distinction is whether the rest state is caused by a collision/capture outcome or by friction-mediated deceleration during sliding.

**Sliding arrest versus settling.** `sliding_arrest` marks the end of translational sliding. `settling` marks the final stabilization after residual bouncing, wobbling, rotation, or corrective motion has disappeared. A block that skids across a floor and stops receives `sliding_arrest`; an object that bounces or wobbles before becoming stable receives `settling` when the residual motion ends.

**Rotation arrest versus settling.** `rotation_arrest` is used when visible angular motion stops. `settling` is used when the whole object or system reaches stable equilibrium after all residual motion. A rolling or spinning cylinder can first receive `rotation_arrest`; if it continues to wobble or shift slightly before becoming stable, `settling` may occur later.

**Toppling versus rotation onset.** `rotation_onset` marks the beginning of angular motion. `toppling` is more specific: it marks loss of stability that causes an object to fall over. A block that begins spinning after being clipped may receive `rotation_onset`; a domino that passes its balance point and begins falling receives `toppling`.

**Surface transition versus landing.** `surface_transition` marks a change from one supporting surface or surface condition to another, such as smooth-to-rough or ramp-to-floor, when the object is already supported and continues moving. `landing` marks falling-to-supported motion. The key distinction is whether the object was airborne/falling before the transition.

Table 9: **APT taxonomy by example.** Each APT is a visually grounded physical state change. Examples are written as annotation cues: what changes in the video, and why that change is a distinct physical transition.

| # | Label | Before and after | Example annotation cue |
|---|---|---|---|
| **Domain I: Contact & Collision Mechanics** *(regime changes at contact boundaries)* | | | |
| 1 | `contact_init` | Free then contact | A rolling ball first touches a wall; the frame where the visible gap closes is the transition. |
| 2 | `contact_break` | Contact then free | Two blocks separate after being pressed together; the first frame with visible separation marks contact loss. |
| 3 | `elastic_rebound` | Approach then retreat | A ball hits a rigid barrier and reverses direction while retaining most of its speed. |
| 4 | `inelastic_capture` | Moving then at rest | A puck strikes a soft object and stops against it instead of bouncing away. |
| **Domain II: Gravitational & Projectile Dynamics** *(regime changes under gravity)* | | | |
| 5 | `free_fall_onset` | Supported then falling | A cube slides past the edge of a platform; the last supported frame is followed by downward motion. |
| 6 | `projectile_apex` | Ascending then descending | A launched ball rises, pauses near its highest point, and then begins to fall. |
| 7 | `landing` | Falling then supported | A falling object first touches the floor or platform and begins responding to the surface. |
| **Domain III: Friction & Surface Dynamics** *(friction-mediated regime changes)* | | | |
| 8 | `static_breakaway` | Stationary then sliding | A stationary block begins to slide after another object pushes it hard enough to overcome static friction. |
| 9 | `sliding_arrest` | Sliding then stationary | A box skids across a surface and visibly comes to rest. |
| 10 | `surface_transition` | Surface A then surface B | A sliding object crosses from a smooth ramp onto a rough floor and its motion changes. |
| **Domain IV: Rotational & Stability Dynamics** *(rotational and stability regime changes)* | | | |
| 11 | `rotation_onset` | Non-rotating then rotating | A block clipped at one corner begins to spin rather than translate straight forward. |
| 12 | `toppling` | Stable then unstable | A domino tilts past its balance point and begins falling under gravity. |
| 13 | `rotation_arrest` | Rotating then stationary | A spinning cylinder loses angular motion and becomes visually still. |
| 14 | `settling` | Oscillating then equilibrium | After bouncing or wobbling, an object stops making visible corrective motion and remains stable. |

### B.7 Taxonomy by Example

Table 9 expands the compact taxonomy in the main text with example annotation cues. The examples are written as visual-causal cues: what changes in the video, and why that change is a distinct physical transition.

### B.8 Domain-Level Explanation

**Domain I: Contact & collision mechanics.** This domain separates contact boundary changes from the outcomes of contact. `contact_init` and `contact_break` describe whether two objects or an object and a surface are in contact. `elastic_rebound` and `inelastic_capture` describe what the contact does to motion. For example, when a ball hits a wall and bounces back, annotators mark `contact_init` at first touch, `elastic_rebound` at the visible reversal from approach to retreat, and `contact_break` when visible separation appears. When a moving object strikes another object and remains pressed against it, the relevant outcome is `inelastic_capture`: motion is dissipated into rest rather than converted into rebound.

**Domain II: Gravitational & projectile dynamics.** This domain decomposes falling and projectile events into support loss, apex reversal, and regained support. `free_fall_onset` marks the moment when an object becomes unsupported and gravity begins to dominate its motion. `projectile_apex` marks the transition from upward to downward vertical motion. `landing`

marks the falling-to-supported transition. An event-level label such as "falls" merges these phases, while the APT chain records the sequence of mechanism-typed changes along the trajectory.

**Domain III: Friction & surface dynamics.** This domain captures friction-mediated and surface-mediated changes in motion. `static_breakaway` and `sliding_arrest` are opposite transitions: rest becomes sliding after static friction is overcome, and sliding becomes rest after motion is dissipated. `surface_transition` captures a change in supporting surface or surface condition, such as a block crossing from a smooth ramp to a rough floor. The cue is not merely that the object continues moving, but that its motion changes because the surface interaction changes.

**Domain IV: Rotational & stability dynamics.** This domain separates angular motion, loss of stability, and final equilibrium. `rotation_onset` marks the beginning of visible angular motion, while `rotation_arrest` marks the end of angular motion. `toppling` is more specific: it indicates that an object loses stability and begins to fall over. `settling` marks the final damping transition after bouncing, wobbling, rotation, or other residual motion disappears.

### B.9 Label-Level Annotation Notes

**`contact_init`.** Mark the first frame where two previously separated objects, or an object and a surface, visibly make contact. The key cue is closure of the visible gap. Do not use this label for support loss or generic proximity without contact.

**`contact_break`.** Mark the first frame where two previously contacting objects, or an object and a surface, visibly separate. The key cue is appearance of a visible gap. If the separation causes gravity-driven falling, `free_fall_onset` may be the more explanatory label for the same object.

**`elastic_rebound`.** Mark the visible reversal from approach to retreat after contact. The transition is not merely that contact exists, but that collision response redirects motion. This label is appropriate when the object retains substantial motion after impact rather than being captured.

**`inelastic_capture`.** Mark the moment when a moving object collides and is effectively captured or stopped by the contact. The key cue is that motion is dissipated into rest at or against another object rather than transformed into rebound. This label is contact-outcome specific, not a general sliding-to-rest label.

**`free_fall_onset`.** Mark the moment when an object loses support and begins gravity-driven falling. Examples include a block sliding off a platform or an object released from a support. The cue is not just downward motion, but downward motion after support is removed.

**`projectile_apex`.** Mark the highest point of a launched or bouncing trajectory, where vertical motion changes from ascending to descending. The object may appear momentarily slow or stationary in the vertical direction. This label is used for projectile-like motion, not for arbitrary pauses.

**`landing`.** Mark the transition from falling or airborne motion to being supported by a surface. The cue is first surface response after downward motion, such as stopping, bouncing, sliding, or becoming supported. If the object is not airborne before the transition, use another contact or surface label.

**`static_breakaway`.** Mark the moment a previously stationary object begins to slide because applied force overcomes static friction. The key cue is rest-to-sliding under interaction. Do not use this label for an object that is already moving.

**`sliding_arrest`.** Mark the moment a sliding object visibly comes to rest. The key cue is sliding-to-stationary motion, usually due to friction. If the object continues to wobble after translational motion stops, `settling` may occur later.

Table 10: **Example event labels and possible APT chains.** Event labels describe clip-level outcomes, while APT chains expose the physical transitions that make those outcomes unfold.

| Coarse event label | Possible APT chain | Causal interpretation |
|---|---|---|
| Ball bounces on a table | `contact_init` → `elastic_rebound` → `contact_break` → `landing` → `settling` | Contact activates collision response; rebound reverses motion; later support and damping stabilize the object. |
| Block falls off a platform | `free_fall_onset` → `landing` → `sliding_arrest` | Support loss makes gravity dominate; surface contact restores support; friction dissipates sliding. |
| Puck hits a soft obstacle | `contact_init` → `inelastic_capture` → `settling` | Contact begins, impact energy is dissipated rather than rebounded, and the system stabilizes. |
| Domino falls | `toppling` → `rotation_onset` → `settling` | Loss of stability induces angular motion; damping and contact eventually bring the object to equilibrium. |
| Box starts sliding and stops | `static_breakaway` → `sliding_arrest` | Applied force overcomes static friction; kinetic friction dissipates motion into rest. |
| Object crosses surfaces | `surface_transition` → `sliding_arrest` | The support condition changes, altering motion; friction on the new surface eventually stops the object. |

**`surface_transition`.** Mark the moment an object crosses from one supporting surface or surface condition to another. Examples include ramp-to-floor, smooth-to-rough, or low-friction-to-high-friction regions. The transition should be accompanied by a visible or expected change in motion caused by the new surface condition.

**`rotation_onset`.** Mark the moment an object begins visible angular motion. This may occur when an off-center impact produces torque or when an object begins rolling or spinning. If the angular motion is caused by loss of stability and falling over, `toppling` may be more specific.

**`toppling`.** Mark the moment a previously stable object loses stability and begins to fall over. The key cue is crossing from stable upright posture into an unstable falling regime. This label is common in domino-like sequences or tall-object imbalance.

**`rotation_arrest`.** Mark the moment visible angular motion stops. The object may still translate slightly or wobble afterward; in that case, `settling` may occur later. This label specifically refers to angular motion ending.

**`settling`.** Mark the final stabilization after residual motion disappears. Residual motion may include bouncing, wobbling, rocking, small corrective translation, or remaining rotation. This label is used when the object or system becomes stably at rest.

### B.10 APT Chains by Example

Table 10 illustrates how coarse event labels are expanded into APT chains. The purpose is not to prescribe a unique chain for every possible video, but to show how event names become mechanism-typed transition sequences.

### B.11 APT Labels as Simulation Targets

The APT definition serves two roles. First, it tells annotators what transition to mark in an existing video. Second, it provides a label-first specification for simulation. Once an APT label or chain is fixed, a simulator can be configured to instantiate that transition by controlling geometry, mass, initial velocity, applied force, friction, restitution, support, and layout. For example, an `elastic_rebound` scene can be produced by setting a rigid barrier and sufficient restitution; a `sliding_arrest` scene can be produced by setting an initial sliding velocity and frictional

surface; a `toppling` scene can be produced by applying an off-center push to a tall object near its stability boundary.

This label-first route is important because APTs need not be discovered passively. Rare transitions can be generated by specifying the desired causal pattern before rendering the scene. Thus the same label space functions as an annotation unit, an evaluation target, a simulation specification, and a training supervision signal.

### B.12 Mechanism-Conditioned Decoding

APT prediction can be formulated as mechanism-conditioned decoding. Instead of mapping video evidence directly to one of all transition labels, the model first identifies the active physical domain and then decodes the transition type within that domain:

$$p(\ell_i \mid V, \tau_i) = \sum_{d_i} p(\ell_i \mid d_i, V, \tau_i)\, p(d_i \mid V, \tau_i). \tag{6}$$

This factorization reflects the mechanism-centered organization of the taxonomy. Support loss and downward motion point to the gravity domain before `free_fall_onset`; gap closure points to the contact domain before `contact_init`; approach-to-retreat reversal after impact points to collision response before `elastic_rebound`; sliding-to-rest motion points to friction before `sliding_arrest`; loss of balance points to stability before `toppling`.

This structure is useful for learning because it discourages pure label memorization. The model must associate a visible cue with a physical mechanism and then map that mechanism to a transition type. The reported experiments do not implement this factorization; APT-Tune uses the flat 14-type prompt of Listing 16, and mechanism-conditioned decoding remains future work.

### B.13 Summary of the Supplementary Definition

APTs are defined to capture the physical transitions that event-level labels collapse. They are visually grounded, vocabulary-level atomic, and tied to physical mechanisms. A video is represented as an ordered APT chain, but the chain can be expressed in multiple formats, including JSON, QA, multiple-choice, and natural-language description. The taxonomy is scoped to visually grounded macroscopic mechanics, where transitions are both physically meaningful and visually localizable. This makes APTs suitable not only for diagnosis, but also for simulation construction and format-robust transition fine-tuning.

## C Data Construction and Implementation Details

This section details the construction workflow summarized in Figure 3. We use human annotations as the perceptual validation anchor, `qwen/Qwq-32B` as the text-only LLM for category-level scene proposal and GT-trace parsing, and `Qwen3.5-VL` for post-render semantic grounding. The following subsections provide the coupling rules, prompts, simulator conversion, and human validation procedure.

### C.1 Human-Labeled Validation Anchor

We use human annotation as the perceptual anchor for APT construction. Although simulation provides scalable state traces, APTs are visually grounded physical transitions, so their boundaries must match what humans can reliably perceive in rendered video. We annotate controlled physical videos from CLEVRER (Yi et al., 2019) and Physion++ (Bear et al., 2021; Tung et al., 2023): annotators watch each clip, mark visible physical state changes, assign one of the 14 APT types, and record transition timestamps. The human-labeled controlled-video anchor contains 500 CLEVRER source trials with 6,316 APTs, covering 7 of 14 types, and 746 Physion++ trials with 20,987 APTs from six rigid-body scenario families, yielding 1,246 trials, 27,303 APTs, and full 14-type coverage. Rule-guided simulator construction adds 1,254 trials and approximately 27,700 APTs, for a combined dataset of 2,500 trials and approximately 55,000 APTs. We use this anchor to calibrate coupling rules, prompts, YAML success checks, simulator-trace parsing rules, and motion/contact thresholds.

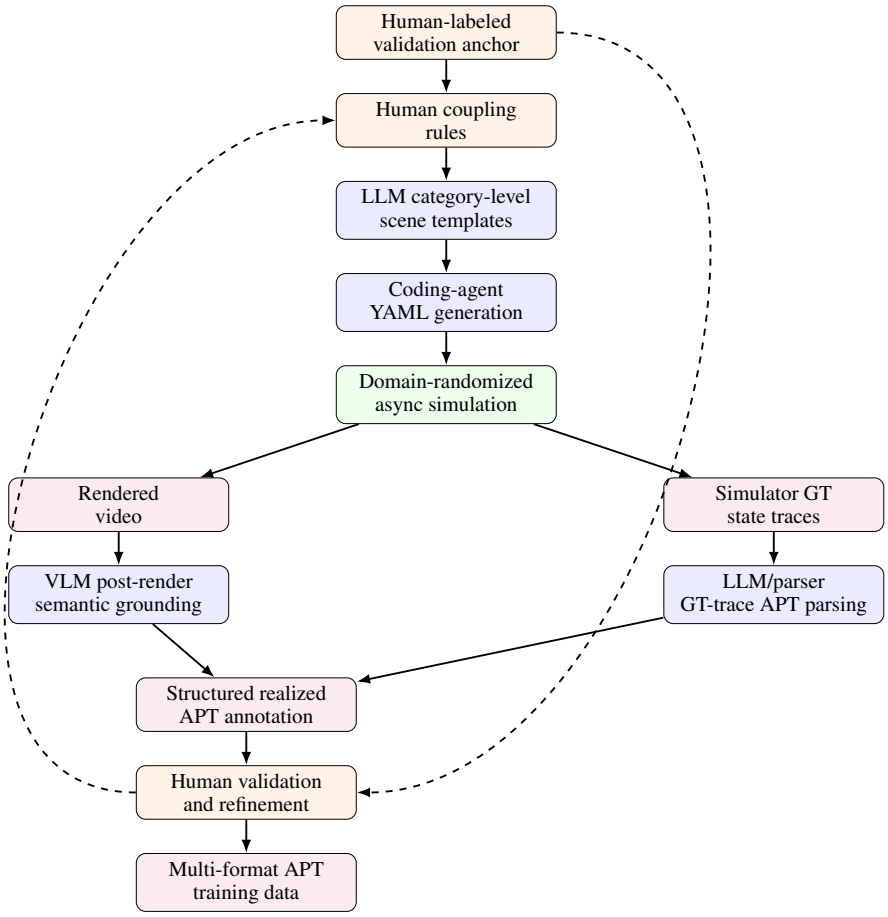


Figure 5: **Rule-guided coarse-to-fine APT construction.** Humans provide a perceptual anchor and admissible causal couplings. An LLM proposes category-level templates, a coding agent converts them to executable YAML, and asynchronous simulation executes them under domain randomization. After rendering, a VLM grounds the concrete visible objects, while an LLM-assisted parser reads simulator GT traces to recover precise APT stages. Human validation refines prompts, templates, thresholds, and success checks, yielding multi-format supervision.

### C.2 Rule-Guided Coarse-to-Fine Pipeline

APT types are atomic, but they are not freely composable. Randomly combining transition labels often produces physically incoherent or visually unnatural scenes. For example, gravity-driven falling naturally couples with landing, contact initiation, rebound, contact break, and settling; friction naturally couples with sliding and arrest; and stability loss naturally couples with rotation, toppling, landing, and settling. We therefore use a rule-guided construction process rather than random APT sampling. Humans first define admissible causal couplings; an LLM expands them into category-level scene templates under simulator constraints; a coding agent converts templates into executable YAML; asynchronous domain-randomized simulation produces rendered videos and state traces; a VLM grounds the rendered appearance; and an LLM-assisted parser converts simulator traces into realized APT chains.

### C.3 Step 1: Human Coupling Rules

Humans first specify coarse physical couplings rather than complete videos or frame-level labels. Each rule describes which APT domains can naturally co-occur and in what approximate temporal order. This prevents the generator from sampling arbitrary APT sequences that have no plausible physical realization.

Listing 1: Demo human-defined coupling rule.

```
Allowed domains:
  friction, contact, gravity, settling

Allowed APT types:
  static_breakaway
  surface_transition
  sliding_arrest
  free_fall_onset
  contact_init
  landing
  inelastic_capture
  settling

Physical sense:
  A dynamic rigid object can slide on a support surface, cross into a rougher region, and stop
      due to friction.
  A garment can become unsupported, fall under gravity, contact the rigid object or support
     surface, and settle without rebound.
  These transitions are physically compatible and can form a longer APT chain than a simple
     single-event clip.

Simulator asset classes allowed:
  dynamic_rigid, static_rigid, articulated, garment, fluid, sand
```

### C.4 Step 2: LLM Category-Level Scene Template Proposal

Given a human coupling rule, `qwen/Qwq-32B` proposes category-level scene templates. The LLM is not asked to choose final assets or annotate timestamps. Instead, it proposes a causal scene skeleton under simulator constraints: which object classes participate, which interactions should occur, and which APT domains are expected.

Listing 2: Prompt for LLM category-level scene-template proposal.

```
You are helping construct simulation scenes for Atomic Physical Transitions (APTs).
The goal is to create physically plausible scenes that contain multiple coupled APTs.

Supported simulator asset classes:
  dynamic_rigid
  static_rigid
  articulated
  garment
  fluid
  sand

Human-defined coupling rule:
  Friction, contact, gravity, and settling may co-occur.
  A dynamic rigid object may slide on a support surface and stop due to friction.
  A garment may fall onto the rigid object or support surface and then settle.

Target APT domains:
  friction, contact, gravity, settling

Instructions:
  Propose 3 category-level scene templates.
  For each template, output:
    (1) scene name,
    (2) object slots and their simulator asset classes,
    (3) natural-language causal scene skeleton,
    (4) expected APT types,
    (5) success checks,
    (6) likely failure cases.

Do not choose final object identity, color, or material.
Do not output timestamps.
```

Listing 3: Demo LLM answer: category-level templates.

```
Template 1: rigid-slide-garment-cover

Object slots:
  support: static_rigid, e.g., table or platform
  mover: dynamic_rigid, e.g., cube, sphere, cylinder, mug, or apple
  cover: garment, e.g., pants, shorts, or cloth
```

```
Causal scene skeleton:
  A dynamic rigid object starts on a support surface, receives an initial horizontal velocity,
  slides across the surface, crosses into a higher-friction region, and stops.
  A garment is released above the rigid object, falls under gravity, contacts the rigid object
  or the support surface, and settles.

Expected APTs:
  static_breakaway
  surface_transition
  sliding_arrest
  free_fall_onset
  contact_init
  landing
  inelastic_capture
  settling

Success checks:
  The rigid object should move tangentially for at least 0.3 s and later remain below a
     velocity threshold.
  The garment should have a clear downward motion phase, create a new contact pair, and remain
      approximately at rest.

Failure cases:
  The rigid object never moves.
  The garment misses the support area.
  The garment penetrates the table.
  The garment remains airborne at the end.
  The rigid object rebounds instead of stopping.

Template 2: rigid-bump-then-cloth-damping

Object slots:
  support: static_rigid
  mover: dynamic_rigid
  obstacle: static_rigid or dynamic_rigid
  cover: garment

Causal scene skeleton:
  A dynamic rigid object slides into a low obstacle, changes direction or starts rotating,
  slows on a rough surface, and is later covered by a falling garment that damps residual
     motion.

Expected APTs:
  static_breakaway
  contact_init
  rotation_onset
  sliding_arrest
  free_fall_onset
  landing
  settling

Template 3: falling-rigid-with-garment-capture

Object slots:
  support: static_rigid
  falling_object: dynamic_rigid
  cover: garment

Causal scene skeleton:
  A small dynamic rigid object falls from above a support surface, lands with low restitution,
  slides briefly, and is later covered by a falling garment.

Expected APTs:
  free_fall_onset
  landing
  contact_init
  inelastic_capture
  sliding_arrest
  settling
```

### C.5 STEP 3: CODING-AGENT YAML GENERATION

A coding agent converts the selected category-level scene template into an executable YAML scene. This step makes the template simulatable and introduces diversity in initial conditions. The YAML specifies object slots, candidate assets, randomized positions, initial velocities, material ranges, friction and restitution ranges, camera and lighting settings, success checks, and trace logging. All asset paths below are anonymized.

Listing 4: Prompt for coding-agent YAML generation.

```
Convert the following category-level APT template into an executable YAML scene.

Template name:
  rigid-slide-garment-cover

Scene skeleton:
  A dynamic rigid object slides on a table, crosses into a rougher region,
  slows due to friction, and stops.
  A garment is released above the rigid object, falls onto it or the table,
  and settles.

Available anonymized asset roots:
  Tables:
    ~/Desk/...
  Dynamic rigid objects:
    ~/Object/Mug
    ~/Object/apple
    ~/Object/Sugar_Box
  Garments:
    ~/Object/Garment/Pants
  Materials:
    ~/Material/...
  Backgrounds:
    ~/Background/...

Requirements:
  Use domain randomization over asset identity, color, material, initial pose,
  initial velocity, friction, restitution, lighting, and camera.
  Record per-frame pose, linear velocity, angular velocity, contact pairs,
  support relation, object identity, material parameters, and surface region ID.
  Add success checks for sliding motion, surface transition, arrest, garment falling,
  garment contact, and final settling.
  Return YAML only.
```

Listing 5: Demo coding-agent YAML: friction-contact-garment scene.

```yaml
scene_name: rigid_slide_garment_cover
description: >
  A dynamic rigid object slides on a table, enters a higher-friction region,
  stops, and is later covered by a falling garment.

background:
  intensity: [1000, 2000]
  texture:
    - "~/Background/champagne_castle_1_4k.hdr"
    - "~/Background/evening_road_01_4k.hdr"
    - "~/Background/kloofendal_48d_partly_cloudy_4k.hdr"
    - "~/Background/kloppenheim_02_4k.hdr"
    - "~/Background/mealie_road_4k.hdr"
    - "~/Background/moonlit_golf_4k.hdr"
    - "~/Background/noon_grass_4k.hdr"
    - "~/Background/qwantani_4k.hdr"

light:
  light_1:
    types: ["Rect", "Sphere", "Cylinder"]
    common:
      initial_pos_range: [-0.2, -0.2, 0.0, 0.2, 0.5, 0.0]
      initial_ori_range: [-30, -30, -30, 30, 30, 30]
      color_temperature_range: [3000, 9000]
      color_range: [[0.8, 0.8, 0.8], [1.2, 1.2, 1.2]]
    Rect:
      intensity_range: [2000, 5000]
      width_range: [1.0, 3.0]
      height_range: [1.0, 3.0]
    Sphere:
      intensity_range: [5000, 10000]
      radius_range: [0.1, 0.5]
    Cylinder:
      intensity_range: [4000, 8000]
      radius_range: [0.1, 0.3]
      length_range: [1.0, 2.0]

camera:
  random: true
  look_at: [0.0, 0.2, 0.75]
  pos_range: [-1.8, -2.2, 1.2, 1.8, -1.2, 2.2]
  focal_length_range: [24, 45]
```

```yaml
objects:
  support_table:
    usd:
      - "~/Desk/Collected_appleseed_coffeetable/appleseed_coffeetable_inst_base.usd"
      - "~/Desk/Collected_cline/cline_inst_base.usd"
      - "~/Desk/Collected_oaktablesmall/oaktablesmall_inst_base.usd"
      - "~/Desk/Collected_willowbench/willowbench_inst_base.usd"
    random: true
    default_material: true
    num_per_env: 1
    semantic_label: "static_support_surface"
    support_table_1:
      pos: [0.0, 0.0, 0.0]
      ori: [0.0, 0.0, 90.0]
      visual:
        scale: [2.0, 2.0, 1.0]
        color:
          - [0.8, 0.7, 0.6]
          - [0.6, 0.5, 0.4]
          - [0.4, 0.3, 0.2]
          - [0.2, 0.1, 0.0]
        visual_material:
          mdl_folder: "~/Material/Base/Textiles"
      physics:
        type: "geometry"
        collision: true
        surface_regions:
          low_friction_region:
            x_range: [-0.70, -0.05]
            static_friction_range: [0.05, 0.20]
            dynamic_friction_range: [0.05, 0.20]
          high_friction_region:
            x_range: [-0.05, 0.70]
            static_friction_range: [0.80, 1.30]
            dynamic_friction_range: [0.70, 1.20]

  mover:
    common:
      primitive_scale: [0.30, 0.30, 0.30]
      initial_pos_range: [-0.60, -0.05, 0.75, -0.45, 0.20, 0.82]
      initial_ori_range: [-10, -10, -20, 10, 10, 20]
    usd:
      - "~/Object/Mug"
      - "~/Object/apple"
      - "~/Object/Sugar_Box"
    random: true
    num_per_env: 1
    semantic_label: "dynamic_rigid_mover"
    mover_1:
      visual:
        scale: [1.0, 1.0, 1.0]
        visible: true
        color:
          - [1.0, 0.0, 0.0]
          - [0.0, 0.0, 1.0]
          - [1.0, 1.0, 0.0]
          - [0.0, 1.0, 1.0]
        visual_material:
          mdl_folder: "~/Material/Base/Textiles"
      physics:
        type: "dynamic"
        collision: true
        mass_range: [0.2, 0.8]
        ratio: 1.1
        physics_material:
          static_friction_range: [0.10, 0.35]
          dynamic_friction_range: [0.10, 0.35]
          restitution_range: [0.0, 0.10]
        linear_velocity_range:
          - [0.9, -0.05, 0.0]
          - [1.4, 0.05, 0.0]
        angular_velocity_range:
          - [0.0, 0.0, -0.5]
          - [0.0, 0.0, 0.5]
        track_contact_forces: true
        prepare_contact_sensor: true

  garment_cover:
    common:
      primitive_scale: [0.50, 0.50, 0.50]
```

```
      initial_pos_range: [0.05, -0.10, 1.35, 0.35, 0.25, 1.70]
      initial_ori_range: [-15, -15, 60, 15, 15, 120]
    usd:
      - "~/Object/Garment/Pants"
    random: true
    num_per_env: 1
    semantic_label: "falling_garment"
    garment_cover_1:
      visual:
        scale: [1.0, 1.0, 1.0]
        visible: true
        color:
          - [1.0, 0.0, 0.0]
          - [0.0, 1.0, 0.0]
          - [0.0, 0.0, 1.0]
          - [1.0, 1.0, 0.0]
          - [1.0, 0.0, 1.0]
          - [0.0, 1.0, 1.0]
        visual_material:
          mdl_folder: "~/Material/Base/Textiles"
      physics:
        type: "garment"
        collision: true
        ratio: 1.1
        release_time_range: [0.55, 0.90]
        particle_system:
          particle_system_enabled: true
          enable_ccd: true
          solver_position_iteration_count: 16
          max_depenetration_velocity: null
          global_self_collision_enabled: true
          non_particle_collision_enabled: true
          contact_offset: 0.010
          rest_offset: 0.006
          particle_contact_offset: 0.010
          fluid_rest_offset: 0.006
          solid_rest_offset: 0.006
        particle_material:
          adhesion: 10
          adhesion_offset_scale: 0.0
          cohesion: 0.1
          particle_adhesion_scale: 3.0
          particle_friction_scale: 3.0
          drag: 0.0
          lift: 0.0
          friction: 10.0
          damping: 5.0
          gravity_scale: 1.0
        garment_config:
          particle_mass: 5e-10
          self_collision: true
          self_collision_filter: true
          stretch_stiffness: 1e12
          bend_stiffness: 1e3
          shear_stiffness: 1e3
          spring_damping: 5.0

trace_logging:
  fps: 60
  fields:
    - pose
    - linear_velocity
    - angular_velocity
    - contact_pairs
    - contact_forces
    - support_relation
    - object_identity
    - material_parameters
    - surface_region_id

success_checks:
  intended_domains:
    - friction
    - contact
    - gravity
    - settling
  required_conditions:
    mover_slides:
      object: "mover"
      min_horizontal_displacement: 0.25
      min_duration_sec: 0.25
```

```
    surface_transition:
      object: "mover"
      from_region: "low_friction_region"
      to_region: "high_friction_region"
    sliding_arrest:
      object: "mover"
      speed_below: 0.03
      hold_duration_sec: 0.20
    garment_falls:
      object: "garment_cover"
      min_downward_displacement: 0.30
    garment_contacts:
      object: "garment_cover"
      target_any_of: ["mover", "support_table"]
    final_settling:
      objects: ["mover", "garment_cover"]
      linear_speed_below: 0.03
      angular_speed_below: 0.10
      hold_duration_sec: 0.30
  failure_filters:
    - "garment misses table and falls out of view"
    - "mover never enters high-friction region"
    - "mover leaves support surface"
    - "simulation contains severe interpenetration"
```

### C.6 Step 4: Domain-Randomized Asynchronous Simulation

We execute each YAML scene in Isaac Lab (Nvidia, 2025) with asynchronous parallel simulation and domain randomization. The same category-level template can produce many concrete videos because asset identity, color, material, physical parameters, initial pose, initial velocity, camera, and lighting are sampled at simulation time. For example, the abstract slot `dynamic_rigid_mover` may render as a blue cube, a red mug, a yellow cylinder, or a green apple; the `falling_garment` slot may render as red pants, blue pants, or a cloth-like garment. This design increases both visual diversity and APT-coupling diversity while preserving the intended causal structure.

The simulator records two synchronized outputs: (1) rendered video frames and (2) frame-level ground-truth state traces, including object identity, pose, linear velocity, angular velocity, contact pairs, contact forces, support relations, material parameters, and surface-region identifiers. The intended coupling rule is only a causal prior. The final labels are assigned from the realized trace, so failed executions can be filtered, unintended outcomes can be relabeled, and extra realized transitions can be retained when supported by the trace.

### C.7 Step 5: VLM Post-Render Semantic Grounding

Because domain randomization chooses concrete assets during simulation, the abstract template does not know the final visible objects. We therefore use `Qwen3.5-VL` after rendering to ground object slots into concrete visual descriptions. The VLM identifies object category, color, material appearance, location, and visible relations. It is not asked to infer APT types or timestamps.

Listing 6: Prompt for VLM post-render semantic grounding.

```
You are given frames from a rendered physics simulation and the abstract scene template below.

Abstract slots:
  support: a static support surface
  mover: a dynamic rigid object sliding on the support
  cover: a garment released above the mover

Task:
  Identify the concrete visible object corresponding to each slot.
  Return only JSON with:
    slot
    object_category
    color
    material_appearance
    location
    visible_relation

Important:
  Do not infer APT labels.
  Do not infer timestamps.
```

```
  Do not describe invisible simulator states.
  Only ground the abstract template into visible objects and relations.
```

Listing 7: Demo VLM answer: semantic grounding.

```
{
  "support": {
    "object_category": "coffee table",
    "color": "brown",
    "material_appearance": "wood-like",
    "location": "center of the scene",
    "visible_relation": "supports the moving object and later the garment"
  },
  "mover": {
    "object_category": "cube",
    "color": "blue",
    "material_appearance": "matte rigid object",
    "location": "on top of the table",
    "visible_relation": "slides from left to right across the table"
  },
  "cover": {
    "object_category": "pair of pants",
    "color": "red",
    "material_appearance": "soft textile garment",
    "location": "above the table, then on or near the cube",
    "visible_relation": "falls downward and contacts or covers the cube"
  }
}
```

### C.8 Step 6: LLM/Parser GT-Trace-Based APT Stage Parsing

Precise APT timing is obtained from simulator traces, not from visual guessing. We use `qwen/Qwq-32B` as an LLM-assisted structured parser. The parser receives the APT taxonomy, the human-defined coupling rule, the VLM-grounded object names, and a compact trace summary. It converts numeric state changes into realized APT instances. The intended template is treated as a prior, not a label guarantee.

Listing 8: Prompt for LLM GT-trace-to-APT parsing.

```
You are converting simulator ground-truth traces into Atomic Physical Transition annotations.

Important rules:
  Use simulator traces as the source of timing and physical state.
  Use VLM grounding only for object names and visual descriptions.
  The intended template is a prior, not a label guarantee.
  If the realized dynamics differ from the intended template, relabel according to the trace.
  Output concise evidence, not chain-of-thought.

APT definitions:
  contact_init: free then contact; a new object-object or object-surface contact begins.
  contact_break: contact then free; an existing contact pair separates.
  elastic_rebound: approach then retreat; post-contact relative velocity reverses with rebound.

  inelastic_capture: moving then captured/resting; object remains in contact without rebound.
  free_fall_onset: supported then falling; object becomes unsupported and moves downward.
  projectile_apex: ascending then descending; vertical velocity crosses from positive to
     negative.
  landing: falling then supported; falling object first becomes supported by a surface/object.
  static_breakaway: stationary then sliding; object begins sliding after force overcomes
     static friction.
  sliding_arrest: sliding then stationary; sliding object comes to rest and remains below
     threshold.
  surface_transition: moving object crosses from one surface/material region to another.
  rotation_onset: non-rotating then rotating; angular velocity becomes visible/significant.
  toppling: stable then unstable; object loses stability and begins falling over.
  rotation_arrest: rotating then stationary; angular velocity damps below threshold.
  settling: object motion damps to stable rest after wobble, bounce, sliding, or deformation.

Grounded slots:
  mover = blue cube
  support = brown wooden table
  cover = red pants

Trace summary:
  fps = 60
```

```
  t=0.00s:
    blue_cube supported_by table = true
    blue_cube speed = 0.00 m/s

  t=0.15s:
    blue_cube horizontal impulse applied
    blue_cube speed increases from 0.00 to 1.08 m/s

  t=0.44s:
    blue_cube x crosses surface_region low_friction -> high_friction
    speed begins decreasing more rapidly

  t=0.81s:
    blue_cube speed < 0.03 m/s for 0.20s
    blue_cube remains supported_by table = true

  t=0.88s:
    red_pants release flag changes fixed -> dynamic
    red_pants supported = false
    red_pants vertical velocity becomes negative

  t=1.18s:
    contact_pair(red_pants, blue_cube) changes false -> true
    contact_force normal > threshold
    red_pants vertical velocity magnitude decreases sharply

  t=1.25s:
    red_pants remains in contact with blue_cube/table
    no rebound; relative normal velocity does not become positive

  t=1.70s:
    red_pants particle velocities < threshold for 0.30s
    blue_cube speed remains < threshold

Output format:
  A JSON array.
  Each record must contain:
    type
    t_ms
    objects
    evidence
```

Listing 9: Demo LLM answer: realized APT chain.

```
[
  {
    "type": "static_breakaway",
    "t_ms": 150,
    "objects": ["blue cube"],
    "evidence": "blue cube changes from rest to horizontal sliding after impulse"
  },
  {
    "type": "surface_transition",
    "t_ms": 440,
    "objects": ["blue cube", "brown wooden table"],
    "evidence": "blue cube crosses from low-friction to high-friction surface region"
  },
  {
    "type": "sliding_arrest",
    "t_ms": 810,
    "objects": ["blue cube", "brown wooden table"],
    "evidence": "blue cube speed stays below threshold while supported by table"
  },
  {
    "type": "free_fall_onset",
    "t_ms": 880,
    "objects": ["red pants"],
    "evidence": "red pants become unsupported and acquire downward velocity"
  },
  {
    "type": "contact_init",
    "t_ms": 1180,
    "objects": ["red pants", "blue cube"],
    "evidence": "new contact pair between red pants and blue cube appears"
  },
  {
    "type": "landing",
    "t_ms": 1180,
    "objects": ["red pants", "blue cube"],
```

```
    "evidence": "falling red pants first become supported through contact"
  },
  {
    "type": "inelastic_capture",
    "t_ms": 1250,
    "objects": ["red pants", "blue cube"],
    "evidence": "red pants remain in contact without rebound after impact"
  },
  {
    "type": "settling",
    "t_ms": 1700,
    "objects": ["red pants", "blue cube"],
    "evidence": "garment and cube remain below motion thresholds for the hold window"
  }
]
```

This example illustrates why rule-guided simulation produces more compositionally dense APT data than existing event-centered clips. A conventional falling or bouncing clip may contain only two or three obvious transitions. In contrast, the coupled friction-contact-garment scene produces a longer chain involving sliding, surface transition, arrest, falling, contact, landing, capture, and settling.

### C.9 Additional Demo: Gravity-Contact Rigid-Body Scene

We also construct simpler rigid-body templates to cover gravity, contact, landing, inelastic capture, and settling. The example below follows the same rule-guided pipeline but uses only static rigid supports and dynamic rigid objects.

Listing 10: Prompt for LLM scene-template proposal: gravity-contact rigid bodies.

```
Human coupling rule:
  Gravity, contact, landing, inelastic capture, and settling can co-occur.
  Use only static rigid supports and dynamic rigid objects.
  Construct a category-level scene where multiple small rigid objects fall onto a table
  and may contact one another after landing.

Output:
  object slots
  scene skeleton
  expected APTs
  success checks
  likely failure cases
```

Listing 11: Demo LLM answer: gravity-contact template.

```
{
  "template_name": "multi_rigid_drop_on_table",
  "object_slots": {
    "support": "static_rigid table",
    "falling_object_A": "dynamic_rigid object",
    "falling_object_B": "dynamic_rigid object"
  },
  "scene_skeleton": [
    "Two dynamic rigid objects are initialized above a table.",
    "They fall under gravity.",
    "Each object lands on the table.",
    "One object may roll or slide into the other.",
    "Both objects settle on the table."
  ],
  "expected_apts": [
    "free_fall_onset",
    "landing",
    "contact_init",
    "inelastic_capture",
    "sliding_arrest",
    "settling"
  ],
  "success_checks": [
    "each dynamic object has downward velocity before first table contact",
    "each object creates a new contact pair with the table",
    "at least one post-landing object-object contact occurs or the scene is relabeled",
    "all dynamic objects satisfy final rest thresholds"
  ],
  "likely_failure_cases": [
    "one object misses the table",
    "objects bounce too strongly and leave the view",
```

```
    "objects never contact each other",
    "final settling is not reached within the clip"
  ]
}
```

Listing 12: Demo coding-agent YAML: gravity-contact rigid-body scene.

```
scene_name: multi_rigid_drop_on_table
description: >
  Two dynamic rigid objects fall onto a table, land, may collide with
  each other, and finally settle.

background:
  intensity: [1000, 2000]
  texture:
    - "~/Background/champagne_castle_1_4k.hdr"
    - "~/Background/evening_road_01_4k.hdr"
    - "~/Background/noon_grass_4k.hdr"
    - "~/Background/qwantani_4k.hdr"

light:
  light_1:
    types: ["Rect", "Sphere", "Cylinder"]
    common:
      initial_pos_range: [-0.2, -0.2, 0.0, 0.2, 0.5, 0.0]
      initial_ori_range: [-30, -30, -30, 30, 30, 30]
      color_temperature_range: [3000, 9000]
      color_range: [[0.8, 0.8, 0.8], [1.2, 1.2, 1.2]]
    Rect:
      intensity_range: [2000, 5000]
      width_range: [1.0, 3.0]
      height_range: [1.0, 3.0]
    Sphere:
      intensity_range: [5000, 10000]
      radius_range: [0.1, 0.5]
    Cylinder:
      intensity_range: [4000, 8000]
      radius_range: [0.1, 0.3]
      length_range: [1.0, 2.0]

camera:
  random: true
  look_at: [0.2, 0.4, 0.75]
  pos_range: [-1.5, -2.0, 1.2, 1.5, -1.0, 2.2]
  focal_length_range: [24, 45]

objects:
  simple_desk:
    usd:
      - "~/Desk/Collected_appleseed_coffeetable/appleseed_coffeetable_inst_base.usd"
      - "~/Desk/Collected_cline/cline_inst_base.usd"
      - "~/Desk/Collected_oaktablesmall/oaktablesmall_inst_base.usd"
      - "~/Desk/Collected_willowbench/willowbench_inst_base.usd"
    random: true
    default_material: true
    num_per_env: 1
    semantic_label: "static_support_surface"
    simple_desk_1:
      pos: [0.0, 0.0, 0.0]
      ori: [0.0, 0.0, 90.0]
      visual:
        scale: [1.3, 1.3, 1.0]
        color:
          - [0.8, 0.7, 0.6]
          - [0.6, 0.5, 0.4]
          - [0.4, 0.3, 0.2]
          - [0.2, 0.1, 0.0]
        visual_material:
          material_usd_folder: "~/Material/Garment"
      physics:
        type: "geometry"
        collision: true
        physics_material:
          static_friction_range: [0.5, 1.0]
          dynamic_friction_range: [0.4, 0.9]
          restitution_range: [0.0, 0.15]

  falling_rigid_A:
    common:
      primitive_scale: [0.30, 0.30, 0.30]
```

```
      initial_pos_range: [-0.35, -0.20, 1.30, -0.05, 0.20, 1.80]
      initial_ori_range: [-30, -30, -30, 30, 30, 30]
    usd:
      - "~/Object/Mug"
      - "~/Object/apple"
      - "~/Object/Sugar_Box"
    random: true
    num_per_env: 1
    semantic_label: "falling_rigid_A"
    falling_rigid_A_1:
      visual:
        scale: [1.0, 1.0, 1.0]
        visible: true
        color:
          - [1.0, 0.0, 0.0]
          - [0.0, 0.0, 1.0]
          - [1.0, 1.0, 0.0]
          - [0.0, 1.0, 1.0]
        visual_material:
          material_usd_folder: "~/Material/Garment"
      physics:
        type: "dynamic"
        collision: true
        mass_range: [0.2, 0.7]
        ratio: 1.1
        physics_material:
          static_friction_range: [0.2, 0.6]
          dynamic_friction_range: [0.2, 0.6]
          restitution_range: [0.0, 0.20]
        linear_velocity_range:
          - [0.05, 0.00, 0.00]
          - [0.25, 0.10, 0.00]
        angular_velocity_range:
          - [-0.2, -0.2, -0.2]
          - [0.2, 0.2, 0.2]
        track_contact_forces: true
        prepare_contact_sensor: true

  falling_rigid_B:
    common:
      primitive_scale: [0.30, 0.30, 0.30]
      initial_pos_range: [0.05, 0.00, 1.25, 0.35, 0.30, 1.75]
      initial_ori_range: [-30, -30, -30, 30, 30, 30]
    usd:
      - "~/Object/Mug"
      - "~/Object/apple"
      - "~/Object/Sugar_Box"
    random: true
    num_per_env: 1
    semantic_label: "falling_rigid_B"
    falling_rigid_B_1:
      visual:
        scale: [1.0, 1.0, 1.0]
        visible: true
        color:
          - [0.0, 1.0, 0.0]
          - [1.0, 0.0, 1.0]
          - [0.0, 1.0, 1.0]
          - [1.0, 0.5, 0.0]
        visual_material:
          material_usd_folder: "~/Material/Garment"
      physics:
        type: "dynamic"
        collision: true
        mass_range: [0.2, 0.7]
        ratio: 1.1
        physics_material:
          static_friction_range: [0.2, 0.6]
          dynamic_friction_range: [0.2, 0.6]
          restitution_range: [0.0, 0.20]
        linear_velocity_range:
          - [-0.25, -0.10, 0.00]
          - [-0.05, 0.00, 0.00]
        angular_velocity_range:
          - [-0.2, -0.2, -0.2]
          - [0.2, 0.2, 0.2]
        track_contact_forces: true
        prepare_contact_sensor: true

trace_logging:
  fps: 60
```

```
  fields:
    - pose
    - linear_velocity
    - angular_velocity
    - contact_pairs
    - contact_forces
    - support_relation
    - object_identity
    - material_parameters

success_checks:
  intended_domains:
    - gravity
    - contact
    - friction
    - settling
  required_conditions:
    object_A_falls:
      object: "falling_rigid_A"
      min_downward_velocity: 0.2
    object_B_falls:
      object: "falling_rigid_B"
      min_downward_velocity: 0.2
    table_landing_A:
      contact_pair: ["falling_rigid_A", "simple_desk"]
    table_landing_B:
      contact_pair: ["falling_rigid_B", "simple_desk"]
    final_settling:
      objects: ["falling_rigid_A", "falling_rigid_B"]
      linear_speed_below: 0.03
      angular_speed_below: 0.10
      hold_duration_sec: 0.30
  optional_conditions:
    secondary_contact:
      contact_pair_any:
        - ["falling_rigid_A", "falling_rigid_B"]
  failure_filters:
    - "one rigid object misses the support surface"
    - "object bounces out of camera view"
    - "final settling not reached within clip"
    - "severe interpenetration"
```

### C.10 Step 7: Human Validation and Prompt/Template Refinement

The automatic construction pipeline is validated against the human-labeled anchor. For a subset of simulated clips, humans annotate the visible APT chain directly from the rendered video. We compare human annotations with automatically produced chains using type equality and timestamp tolerance. Disagreements are used to refine four components: (1) coupling rules, when a proposed scene is physically unnatural; (2) YAML success checks, when the intended transition does not reliably occur; (3) VLM grounding prompts, when object identity or relation is ambiguous; and (4) GT-trace parsing thresholds, when simulator state changes occur earlier or later than human-visible transitions. After calibration, the automatic pipeline produces APT chains that closely match human annotations while scaling beyond manual labeling.

Listing 13: Prompt for validation disagreement analysis.

```
You are given:
  (1) a human-labeled APT chain,
  (2) an automatically produced APT chain,
  (3) the simulator trace summary,
  (4) the rendered-video grounding.

Compare the human and automatic chains.
For each mismatch, classify the likely cause as one of:
  template_issue
  success_check_issue
  grounding_issue
  trace_threshold_issue
  human_ambiguity
  extra_realized_transition

Suggest one concrete fix.
Do not change labels unless the simulator trace supports the change.
```

Listing 14: Demo validation answer.

```
[
  {
    "mismatch": "automatic contact_init at 1140 ms; human contact_init at 1180 ms",
    "cause": "trace_threshold_issue",
    "analysis": "contact force is nonzero before contact is visually clear",
    "fix": "increase contact-force threshold or require 2 consecutive frames above threshold"
  },
  {
    "mismatch": "automatic omitted landing for red pants",
    "cause": "success_check_issue",
    "analysis": "parser produced contact_init but did not add landing for an object that was
     falling immediately before first support contact",
    "fix": "if an object has negative vertical velocity before first support contact, add
     landing at the first support contact"
  },
  {
    "mismatch": "automatic detected extra surface_transition",
    "cause": "extra_realized_transition",
    "analysis": "simulator trace shows mover crossed from low-friction to high-friction region
      and the speed decay changed visibly",
    "fix": "keep the label when the motion change is visible; otherwise store it as metadata-
     only"
  }
]
```

### C.11 STEP 8: MULTI-FORMAT APT TRAINING DATA

Each accepted clip is exported into multiple aligned formats. This is important because single-format APT JSON training can make the model an APT-format specialist and degrade ordinary event-level video answering. The same physical scene is therefore represented as: (1) APT JSON, (2) grounded object description, (3) causal transition description, (4) event-level QA, and (5) multiple-choice QA. These are export options for an accepted APT clip. In the reported `APT-Tune` experiments, each of 1,000 selected APT video trials contributes one APT-chain example, while the 1,000 multiple-choice and 1,000 description examples come from SSv2 rather than these aligned exports.

Listing 15: Demo multi-format export for the rigid-slide-garment-cover scene.

```
APT JSON:
[
  {"type": "static_breakaway", "t_ms": 150, "objects": ["blue cube"]},
  {"type": "surface_transition", "t_ms": 440, "objects": ["blue cube"]},
  {"type": "sliding_arrest", "t_ms": 810, "objects": ["blue cube"]},
  {"type": "free_fall_onset", "t_ms": 880, "objects": ["red pants"]},
  {"type": "contact_init", "t_ms": 1180, "objects": ["red pants", "blue cube"]},
  {"type": "landing", "t_ms": 1180, "objects": ["red pants", "blue cube"]},
  {"type": "inelastic_capture", "t_ms": 1250, "objects": ["red pants", "blue cube"]},
  {"type": "settling", "t_ms": 1700, "objects": ["red pants", "blue cube"]}
]

Grounded object description:
  A blue cube slides across a brown wooden table, slows down on a rougher region,
  and stops. A red pair of pants falls from above, lands on the cube, remains in
  contact, and settles.

Causal transition description:
  The rigid object first breaks away from rest and slides. Its motion changes when
  it enters a higher-friction surface region, and friction arrests the sliding.
  The garment then becomes unsupported, falls under gravity, initiates contact with
  the cube, is captured without rebound, and settles.

Event-level QA:
  Q: What happens in the video?
  A: A blue cube slides on a table and stops, then a red garment falls onto it and settles.

Multiple-choice QA:
  Q: Which description best matches the video?
  A. The cube bounces elastically off a wall.
  B. The cube slides to a stop, and a garment falls onto it and settles.
  C. The garment flies upward after hitting the cube.
  D. The table topples over.

  Correct answer: B.
```

### C.12 Dataset Split and Evaluation Protocol

The train/test split is stratified by source and scenario family; the human-labeled partition uses seed 7. The 1,246 human-labeled trials split into 996 training trials (400 CLEVRER and 596 Physion++) and 250 test trials (100 CLEVRER and 150 Physion++). The 1,254 simulator trials split into 1,004 training and 250 test trials. Thus the full APT-BENCH split contains 2,000 training trials and **500 test trials with 10,200 APTs**: 100 CLEVRER, 150 Physion++, and 250 simulator trials, matching Table 3. All reported APT-BENCH model results use this full test set.

Table 11: **Dataset composition and trial split.** APT totals cover all trials of each source; test APTs per clip follow Table 3. The simulator and combined APT totals are approximate.

| Source | Train | Test | Total | APTs (all) | APTs/test clip |
|---|---|---|---|---|---|
| CLEVRER | 400 | 100 | 500 | 6,316 | 12.0 |
| Physion++ | 596 | 150 | 746 | 20,987 | 29.0 |
| Sim Data | 1,004 | 250 | 1,254 | ∼27,700 | 18.6 |
| **All** | **2,000** | **500** | **2,500** | **∼55,000** | **20.4** |

All 14 APT types are represented, but the distribution is naturally imbalanced: `contact_init` and `contact_break` occur frequently in collision-heavy scenes, while `projectile_apex` and `free_fall_onset` require specific geometry. We report aggregate recall and, for Qwen3-VL-2B, precision, F1, and prediction counts.

To enable fair zero-shot and fine-tuned comparison across VLMs, we use a frozen evaluation protocol.

1. **Input normalization.** Each model receives the same 32 uniformly sampled frames per video at source resolution. Models with separate frame-rate conventions receive matched 32-frame samples.
2. **Explicit frame times.** Each frame is preceded by a text marker `[Frame i/32, t=<int>ms] <image>`, where the value of `t` is computed from the frame's native index and the clip's native frame rate. The prompt header additionally states the frame count, clip duration, native frame rate, and valid timestamp range. These markers are ordinary text tokens and are identical across training, inference, and every backbone. Neither Qwen3-VL nor InternVL3.5 is given a family-specific video-timestamp interface, so all models receive equivalent temporal information; the same holds for the commercial APIs, which receive the identical interleaved marker/frame sequence. The model therefore predicts *transition* times conditioned on known frame times rather than inferring frame times from pixel content.
3. **Structured output.** A single shared APT-schema prompt asks the model to enumerate timed APTs as a JSON array of records: `{"type": <label>, "t_ms": <int>}`. The 14 type names are listed verbatim with one-line definitions. No chain-of-thought is requested.
4. **Robust parsing.** Predictions are parsed with a tolerant JSON parser that recovers from common formatting errors, including trailing commas, unterminated arrays, and missing closing braces.
5. **Matching.** A predicted APT matches a ground-truth APT if the type matches and the timestamp lies within $\Delta t = \pm 200$ ms. Matching is one-to-one and greedy by confidence when confidence is available, or by parsing order otherwise. Each ground-truth APT can be matched at most once.
6. **Metrics.** We report recall, precision, and F1 under this one-to-one type-and-time matching, together with the mean number of predicted transitions per clip and the median timing error $\Delta T$ over matched events. The auxiliary Type score is type-only recall: for each clip and label, count at most the smaller of the predicted and ground-truth instance counts, disregarding timestamps; sum these counts over clips and labels, then divide by the total number of ground-truth APTs. Thus Type uses the same denominator as recall and is not accuracy conditional on type-and-time matched events. At inference time predictions occasionally truncate mid-JSON when the max-token limit is reached; the tolerant parser recovers any complete leading APT records and discards the remainder rather than dropping the whole prediction, which affects fewer than 3% of predictions.

7. **Controlled comparison.** The same prompt and frame list are used for zero-shot baselines and fine-tuned models. Within each backbone, the zero-shot and tuned evaluations differ in model weights; across backbones, model-specific processors and architectures also differ.

### C.12.1 Temporal Sampling and Frame-Grid Coverage

The default evaluation uses 32 frames. The current 16/32/64-frame comparison is reported in Table 16: grid coverage at $\pm 200$ ms is 61.8%, 95.6%, and 99.7%, respectively. A temporal matching tolerance does not itself measure whether a transition is visually observable in that input. For ground-truth transition times $\{\tau_i\}_{i=1}^{K}$ and sampled native-frame times $\{t_j\}_{j=1}^{F}$, define grid coverage at tolerance $\epsilon$ by

$$C(F, \epsilon) = \frac{1}{K} \sum_{i=1}^{K} \mathbf{1}\left[ \min_{1 \le j \le F} |\tau_i - t_j| \le \epsilon \right]. \tag{7}$$

This diagnostic assumes oracle-correct types and measures the fraction of reference timestamps lying near a sampled frame. It is a timestamp-grid constraint, not visual-evidence coverage: a nearby frame may not reveal the before/after change, and a model may infer a transition time between sampled frames.

The nominal interval for the current protocol is $T/32$: approximately 160 ms for CLEVRER, 403 ms at the mean Physion++ duration, and 223 ms at the mean simulator duration, as in Table 3. For the longest reported Physion++ clip (21.87 s), it is approximately 683 ms. Exact intervals depend on native-frame selection and rounding. Thus, 32 frames reduce temporal gaps relative to 16 but do not guarantee visibility of brief transitions. The current frame-budget comparison establishes high timestamp-grid coverage at 32 and 64 frames and a persistent tuning gain at each tested budget. A tolerance sweep on the current predictions and evaluation conditioned on visible transitions would further separate timing and observation limits.

Listing 16: Evaluation-time APT-schema prompt.

```
You are given 32 frames sampled uniformly from a physics video.

Task:
  Enumerate all visible Atomic Physical Transitions (APTs) in temporal order.
  Return a JSON array.
  Each item must contain:
    type: one of the 14 APT labels
    t_ms: approximate transition time in milliseconds

Allowed APT labels:
  contact_init: free then contact
  contact_break: contact then free
  elastic_rebound: approach then retreat after collision
  inelastic_capture: moving object contacts and remains captured/resting
  free_fall_onset: supported then falling
  projectile_apex: ascending then descending
  landing: falling then supported
  static_breakaway: stationary then sliding
  sliding_arrest: sliding then stationary
  surface_transition: object moves from one surface/material region to another
  rotation_onset: non-rotating then rotating
  toppling: stable then unstable/falling over
  rotation_arrest: rotating then stationary
  settling: motion damps to stable rest

Output rules:
  Return JSON only.
  Do not explain.
  Do not include labels that are not visible.
  Use milliseconds for t_ms.

Example output:
[
  {"type": "free_fall_onset", "t_ms": 420},
  {"type": "contact_init", "t_ms": 830},
  {"type": "elastic_rebound", "t_ms": 910}
]
```

### C.13 APT-TUNE IMPLEMENTATION AND TRAINING DETAILS

`APT-Tune` is a parameter-efficient learning realization of the APT representation. It uses structured SFT to test whether APT supervision improves transition recovery and transfers to other video tasks. Image-pad-aware masking is an implementation requirement for correct multimodal answer-only supervision; format-conditional co-training combines APT chains with event-level rehearsal.

**Optimization.** We use LoRA with rank 16, $\alpha = 32$, and zero dropout on the language-model `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, and `down_proj` modules. The vision encoder and merger are frozen. Training uses bf16 AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$, zero weight decay), an initial learning rate of $10^{-4}$, 30 warm-up steps, and a cosine schedule over three epochs; gradient checkpointing is enabled.

**Implementation requirement: image-pad-aware masking.** The APT chain is a structured target where every token has a clear role (transition type, timestamp, or scaffold), so loss should flow only over the chain. In multimodal SFT, the processor expands each `<image>` placeholder into a grid of `<|image_pad|>` tokens, and a text-only prefix length can leave expanded visual tokens unmasked, turning part of the objective into pad-token reconstruction. We replace $L_{\text{text}}$=len(tokenizer(prompt)) with $L_{\text{mm}}$=len(processor(prompt, images).input_ids), so the loss is computed only over the answer. We stress that this is a statement about the *loss region*, not the input: the interleaved frame-time markers of Appendix C.12 are ordinary text tokens that remain fully visible to the model in both training and inference, and masking never removes them from the context. The two mechanisms therefore act at different stages—timestamp tokens condition the prediction, whereas image-pad-aware masking decides which *target* tokens receive gradient—and Table 15 shows the cost of conflating them: with the text-only prefix length, cross-entropy plateaus at 1.92 because the model is partly trained to reconstruct `<|image_pad|>` tokens.

**Format-conditional co-training.** A specialist trained only on APT-JSON collapses $p(\text{format} \mid \text{prompt})$ onto APT-JSON and loses competence on other tasks (*specialist collapse*). `APT-Tune` therefore co-trains three formats on a single encoder under a shared chat template, dispatched by a record-level `format_type`. The mix is small and balanced: 1,000 APT records (APT-BENCH train + Isaac Lab generated) plus 1,000 MCQ and 1,000 one-sentence-description records drawn from SSv2 (Goyal et al., 2017), an action corpus with no overlap with our zero-shot splits. The non-APT records are not label augmentation, but maintain the format-conditional posterior while the APT objective trains atomic video structure.

## D EXTENDED EXPERIMENTAL RESULTS

### D.1 MVBENCH TRANSFER

Table 12: **MVBench transfer (avg accuracy, %).**

| Backbone | Zero-shot | +APT-Tune |
|---|---|---|
| Qwen3-VL-2B | 51.4 | **53.6** |
| Qwen3-VL-4B | 54.7 | **55.2** |
| Qwen3-VL-8B | 56.8 | **57.6** |
| InternVL3.5-2B | 54.5 | **55.7** |
| InternVL3.5-4B | 55.8 | **56.7** |
| InternVL3.5-8B | 57.1 | **58.3** |

### D.2 SCOPE OF PHYSBENCH TRANSFER

PhysBench is used purely as an out-of-distribution test; no PhysBench data enter `APT-Tune`. The four reported sub-tasks mix physical phenomena and do not isolate transfer outside macroscopic mechanics. Table 13 reports the Qwen3-VL-8B diagnostic under the current 32-frame protocol. Its two five-subtype groups are selected diagnostics rather than an exhaustive partition of PhysBench

or substitutes for the official aggregate score in Table 7. The selected non-mechanics changes are descriptive point estimates and do not establish statistical equivalence. The subtype analysis covers this backbone only; it does not show that all six backbones share the same phenomenon-level pattern.

Table 13: **PhysBench subtypes (%).** Qwen3-VL-8B, 32 frames; group means cover five selected subtypes.

| Subtype | Zero-shot | +APT-Tune | Δ |
|---|---|---|---|
| Motion | 40.9 | 58.4 | +17.5 |
| Viewpoint | 35.5 | 50.7 | +15.2 |
| Manipulation | 38.5 | 59.0 | +20.5 |
| Collision | 41.1 | 62.2 | +21.1 |
| Throwing | 42.4 | 60.1 | +17.7 |
| **Motion/spatial avg.** | **39.7** | **58.1** | **+18.4** |
| Fluid | 44.2 | 44.0 | −0.2 |
| Light | 35.5 | 35.7 | +0.2 |
| Chemistry | 67.3 | 67.0 | −0.3 |
| Temperature | 70.6 | 70.4 | −0.2 |
| Air | 56.4 | 56.2 | −0.2 |
| **Non-mechanics avg.** | **54.8** | **54.7** | **−0.1** |

## D.3 Per-Type Transition Recovery

### D.3.1 Per-Type Precision and Recall at 32 Frames

Table 14 reports the per-type evaluation of Qwen3-VL-2B under the 32-frame protocol, using one-to-one type-and-time matching within ±200 ms. `APT-Tune` improves both recall and precision on all 14 types. Six types with zero zero-shot recall—`free_fall_onset`, `projectile_apex`, `static_breakaway`, `sliding_arrest`, `surface_transition`, and `toppling`—reach 30.8–54.8% recall after tuning. The improvements span all four physical domains, although their magnitude varies by type. Without per-type sample counts or uncertainty estimates, these differences remain descriptive.

Table 14: **Per-type APT results at 32 frames (%).** Qwen3-VL-2B with one-to-one type-and-time matching within ±200 ms. R and P denote recall and precision.

| | Zero-shot | | +APT-Tune | |
|---|---|---|---|---|
| **Type** | **R** | **P** | **R** | **P** |
| `contact_init` | 18.1 | 32.2 | 42.1 | 34.3 |
| `contact_break` | 21.7 | 38.8 | 29.9 | 47.1 |
| `elastic_rebound` | 8.6 | 35.0 | 48.6 | 37.4 |
| `inelastic_capture` | 9.7 | 25.7 | 37.2 | 43.7 |
| `free_fall_onset` | 0.0 | 0.0 | 32.6 | 38.3 |
| `projectile_apex` | 0.0 | 0.0 | 31.3 | 37.4 |
| `landing` | 7.5 | 21.3 | 27.5 | 30.8 |
| `static_breakaway` | 0.0 | 0.0 | 53.1 | 58.6 |
| `sliding_arrest` | 0.0 | 0.0 | 30.8 | 40.6 |
| `surface_transition` | 0.0 | 0.0 | 36.4 | 41.2 |
| `rotation_onset` | 6.9 | 18.6 | 49.7 | 46.3 |
| `toppling` | 0.0 | 0.0 | 54.8 | 45.7 |
| `rotation_arrest` | 5.2 | 20.6 | 36.1 | 57.2 |
| `settling` | 4.2 | 27.4 | 29.3 | 52.9 |

## D.4 Training Recipe Ablations and a Generic Video SFT Baseline

**Learning with APT supervision.** With multimodal answer-only masking, APT-only fine-tuning reaches 33.8% APT recall and 44.7% PhysBench accuracy, compared with 10.0% and 38.5% for the untuned model (Table 15). Adding SSv2 multiple-choice and description records raises these scores

Table 15: **Training recipe ablations and a generic video SFT baseline on Qwen3-VL-2B (%).** The APT-supervised variants vary masking (P1) and format mix (P2); the SSv2-only baseline changes training data and targets.

| Variant | Mask / Format mix | APT recall | SSv2 | PhysBench |
|---|---|---|---|---|
| Base (no SFT) | none | 10.0 | 72.5 | 38.5 |
| Generic video SFT | mm-aware / SSv2-only (no APT) | 11.6 | 78.5 | 40.2 |
| Naive SFT | text-only / APT-only | 23.7 | 50.0 | 36.2 |
| P1 only | mm-aware / APT-only | 33.8 | 68.5 | 44.7 |
| P2 only | text-only / multi | 27.4 | 63.2 | 40.8 |
| **Full `APT-Tune`** | mm-aware / multi | **38.1** | **87.5** | **50.8** |

to 38.1% and 50.8%, respectively, while improving SSv2 accuracy from 68.5% to 87.5%. These comparisons show that training with APT targets improves transition recovery under the evaluated setup, and that the mixed-data recipe further improves performance on the three reported evaluations.

**Comparison with generic video SFT.** The SSv2-only baseline uses 2,000 generic video records and reaches 11.6% APT recall and 40.2% PhysBench accuracy, below the full `APT-Tune` recipe. However, this comparison changes both the training data and the supervision targets: the full recipe additionally includes 1,000 APT-labeled physical videos. We therefore interpret it as a comparison between training recipes, rather than an isolated test of the APT representation. Separating the effects of physical-video exposure, training-set size, and transition structure requires matched-video supervision controls.

The masking ablation diagnoses an incorrect supervision boundary after multimodal token expansion; it is an implementation check rather than evidence for the superiority of APT targets.

**Choice of objective.** Structured SFT jointly supervises type, timestamp, output length, and ordering. A timestamp-only loss requires a separate treatment of missed and mistyped transitions, while timestamp-aware regression and reinforcement-learning objectives remain plausible alternatives. The 38 ms median timing error is conditional on successfully matched Qwen3-VL-2B transitions and does not bound their potential benefit.

### D.5 Frame-Budget Sensitivity

The default 32-frame input can affect whether a brief transition is observed. We evaluate Qwen3-VL-2B at 16, 32, and 64 sampled frames with the same one-to-one type-and-time matching rule (Table 16). The timestamp-grid diagnostic measures the fraction of annotated transition times within $\pm 200$ ms of a sampled frame: coverage rises from 61.8 to 95.6 to 99.7%. Proximity does not guarantee that the before-and-after states are visually resolved.

Zero-shot recall barely changes between 16 and 32 frames (9.8 to 10.0%) despite the large coverage increase, then reaches 18.8% at 64 frames. `APT-Tune` achieves 33.2, 38.1, and 39.7% recall, outperforming the corresponding zero-shot model by 23.4, 28.1, and 20.9 points. The 32-to-64 tuned improvement is 1.6 points even as grid coverage approaches 100%, so timestamp proximity alone does not determine recovery. Grid coverage is not an upper bound on recall: a model may infer a change between frames, while a frame near the timestamp may omit decisive visual evidence.

Table 16: **Frame-budget sensitivity on APT-Bench.** Grid coverage: reference times within $\pm 200$ ms of a sampled frame. Bold marks the 32-frame default.

| Frames | Grid coverage (%) | Zero-shot recall (%) | APT-Tune recall (%) | Gain (pp) |
|---|---|---|---|---|
| 16 | 61.8 | 9.8 | 33.2 | +23.4 |
| **32** | **95.6** | **10.0** | **38.1** | **+28.1** |
| 64 | 99.7 | 18.8 | 39.7 | +20.9 |

# E LIMITATIONS, BROADER IMPACT, FAILURE CASES, AND REPRODUCIBILITY DETAILS

## E.1 LIMITATIONS

**Physics and scene scope.** The 14-type taxonomy focuses on visually grounded macroscopic mechanics, primarily rigid-body and force-mediated contact, collision, gravity, friction, rotation, and stability. Some represented local changes also occur on deformable or fluid-like objects; general fluid, deformable-body, chemical, thermal, optical, and electromagnetic dynamics fall outside this vocabulary. The human-labeled benchmark uses controlled CLEVRER and Physion++ videos, and most clips contain 2–3 APT-relevant interacting objects (at most 5). Dense simultaneous contacts, severe occlusion, and APT localization in unconstrained real-world video remain under-tested. The real-world PhysBench results provide transfer evidence for benchmark answers rather than direct transition-localization evidence.

**Evaluation scope.** APT-BENCH matches transition types and timestamps. It measures recovery of mechanism-typed state changes, while free-form mechanism explanations, links between transitions, interventions, and counterfactuals require different tests. At $\pm 200$ ms, 32 and 64 frames yield 95.6% and 99.7% timestamp-grid coverage, respectively (Table 16); nearby frame timestamps do not guarantee that a brief or occluded change is visually observable. Median timing error is conditional on matched transitions, so it cannot diagnose every unmatched prediction.

**Annotation and metric coverage.** Five trained annotators used independent dual annotation and third-person adjudication, but pre-adjudication records were not retained, preventing an inter-annotator $\kappa$ estimate. The simulator trace parser was calibrated against the human anchor without a separately reported held-out agreement study. Precision, F1, prediction counts, and per-type precision/recall are reported for Qwen3-VL-2B; the broader model panel does not have the same metric coverage or uncertainty estimates. The 32-frame PhysBench subtype diagnostic covers Qwen3-VL-8B and selected phenomena. Small benchmark gains are interpreted descriptively because numerical confidence intervals are not reported.

**Learning objective.** `APT-Tune` uses structured SFT to demonstrate the utility of APT supervision. Timestamp-aware losses and reinforcement-learning objectives may complement it, but they were not compared here. The 38 ms median error on matched Qwen3-VL-2B transitions does not establish the possible gain from a different objective on currently unmatched cases.

## E.2 BROADER IMPACT

Our work has potentially positive impacts in settings that require physically grounded video understanding, including robotics, embodied agents, scientific video analysis, education, and safety-oriented evaluation of multimodal models. By exposing transition-level physical failures that clip-level QA can miss, APT-BENCH can help diagnose brittle reasoning behavior and encourage models that are more faithful to visible mechanism-typed transitions rather than surface correlations.

Potential negative impacts are comparatively limited, because the benchmark focuses on short physical transitions in synthetic and controlled videos rather than personal, medical, or high-stakes human data. Still, improved physical video understanding could indirectly strengthen systems used for automation or surveillance, so we emphasize transparent reporting, controlled benchmarking, and release of evaluation protocols together with limitations. We do not release any high-risk generative model or scraped personal dataset as part of this work.

## E.3 RELEASE, STATISTICAL RELIABILITY, AND COMPUTE DETAILS

**Released assets and documentation.** We release the new APT dataset assets together with annotation guidelines, the multi-format SFT mixture, evaluation scripts, and LoRA training recipes. We also credit the original sources used in our pipeline, including CLEVRER, Physion++, Something-Something v2, PhysBench, Qwen3-VL, and InternVL3.5, and follow the respective public licenses and terms of use of those assets.

**Training and evaluation details.** The main fine-tuning settings, prompts, frame sampling scheme, matching tolerance, and data splits are given in Section 5, Section 6, and Appendix C.12. For the principal fine-tuning comparisons, we report averages over 3 independent runs with different random seeds, tables report run means. The result tables provide point estimates without numerical confidence intervals or bootstrap test outcomes. We therefore interpret small transfer gains descriptively and do not claim statistical significance; reporting variability and paired uncertainty estimates remains necessary.

**Data collection compute.** For simulator-based data construction, each *generated* clip is simulated for approximately 20 s of scene time at roughly real-time speed. This 20 s figure describes the Isaac Lab generation budget only and should not be read as the length of the evaluation clips, which average 8.46 s (Table 3). Using 128 parallel Isaac Lab environments on a single RTX 4090, the idealized wall-clock simulator time is therefore about $20N/128$ seconds for $N$ videos. For the released 1,254-trial split, this corresponds to approximately 196 s of pure simulated time, excluding environment reset, rendering, trace export, and disk I/O overhead; in practice, total collection time is of the same order but somewhat larger.

**Training compute.** All LoRA fine-tuning runs use a single NVIDIA A100 GPU. Representative end-to-end training times are approximately 12 hours for the 2B backbone, 15 hours for the 4B backbone, and 20 hours for the 8B backbone. These numbers cover the reported LoRA training jobs and do not include unsuccessful exploratory runs.