
APT: Atomic Physical Transitions for Causal Video-Language Understanding

Anonymous Author(s)

Affiliation

Address

email

Abstract

Physical events are not understood by their names alone, but by the causal state changes that compose them. A clip-level label such as “bounce” can be correct but hiding the process that makes the event physically valid, from support loss and contact onset to rebound and settling. To make this hidden process explicit, we introduce **Atomic Physical Transitions** (APTs): minimal, temporally localized state changes that bind a visible cue to an active physical mechanism and a before/after dynamical regime. An APT chain represents a video as an ordered causal transition sequence rather than a single aggregate event label: **event labels tell what happened; APT chains explain why it happened**. To make APTs learnable, we construct mixed-source APT data from human annotations and simulator ground truth, covering 14 transition types across contact, gravity, friction, and rotation/stability, with 27,303 timed instances over 1,246 trials. Using this data, we find that current VLMs largely miss transition-level physics, with zero-shot recall at most 14% and errors dominated by missed transitions. Direct fine-tuning on APT chains improves transition detection but causes event-level forgetting, indicating that the model learns a specialized answer format rather than a reusable physical representation. We therefore propose APT-Tune, a parameter-efficient recipe that teaches VLMs to use causal transitions without forgetting how to answer ordinary video questions. It combines image-pad-aware supervision, format-conditional co-training, and mechanism-conditioned domain-to-type decoding to make APT learning format-robust and physically grounded. With only 11 M LoRA parameters on Qwen3-VL-2B, APT-Tune substantially improves APT recall while also improving event-level video transfer. These results show that APTs are not a new answer format, but a human-aligned causal supervision signal for physical video understanding.

1 Introduction

Recall a simple lesson from introductory mechanics: a ball bouncing on a table is not understood by the word “bounce” alone. We learn the event by unpacking its causes: gravity accelerates the ball downward, contact with the table begins, collision forces reverse its motion, friction dissipates residual motion, and the system eventually settles. This is how mechanics is typically organized in instruction: not as a list of event names, but as reusable principles such as force, contact, friction, collision, rotation, equilibrium, and gravitation [24, 42]. Cognitive studies of physics expertise similarly show that experts represent problems by underlying physical principles rather than surface features [10, 21], while event cognition suggests that humans segment continuous activity at meaningful state boundaries [43]. These observations point to a causal view of physical video understanding: to understand an event, a model should recover when the active physical mechanism changes, why that transition occurs, and how it enables the next state.

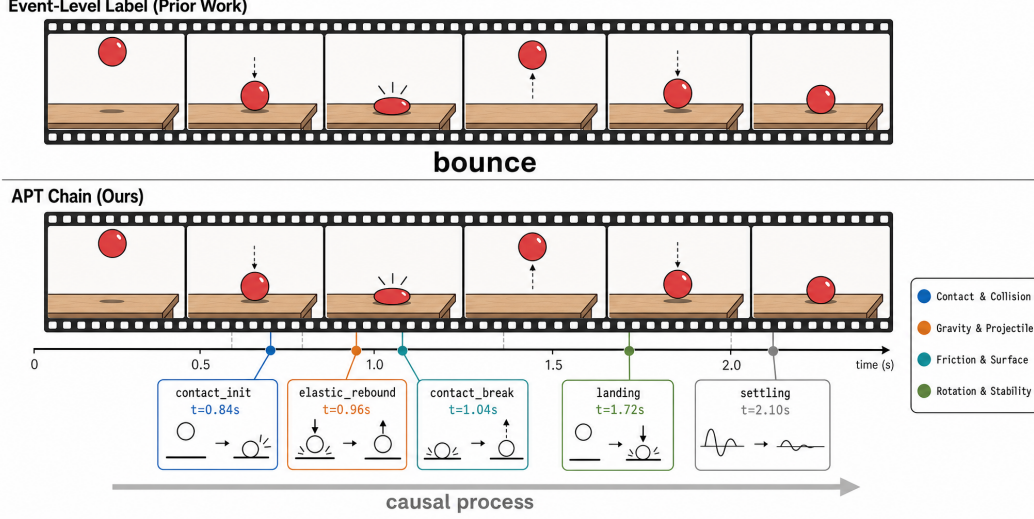


Figure 1: Event-level comparison between standard event-level evaluation and APT.

Existing physics-oriented video benchmarks have made important progress in evaluating object interactions, physical prediction, counterfactual reasoning, and law-aware VQA or generation [41, 5, 37, 9, 33, 30, 38, 28]. However, their supervision and scoring are still largely tied to clip-level outcomes, such as selecting an answer, predicting an event, judging plausibility, or evaluating physical consistency. Such targets are useful but leave the causal process underdetermined: a model may recognize what happened without identifying the intermediate state changes that make the event physically valid. We argue that physical video understanding should expose this hidden process structure through an ordered chain of temporally localized transitions, each linking a visible state change to its active mechanism. These causal transition chains provide a missing intermediate representation between clip-level event recognition and full physical simulation.

To operationalize this view, we introduce **Atomic Physical Transitions (APTs)**: minimal, visually grounded state changes in which a physical system moves between dynamical regimes. APTs are not merely finer event labels, but causal reasoning units that bind a visible cue to an active mechanism and a before/after state. A video is therefore represented as an ordered *APT chain*, i.e., typed timestamps that recover the causal process hidden inside a clip-level event label. Because APTs are defined by physical mechanism rather than scene appearance, the same schema can support annotation, evaluation, simulation specification, and training supervision.

APT-level supervision is difficult to obtain from passive annotation alone, since each clip may contain multiple brief transitions whose boundaries require physical judgment. We therefore build APT data with a mixed-source, human-calibrated pipeline. Human annotators label real-world recordings and existing datasets, including CLEVRER [41], to anchor the 14-label semantics. To scale the data, we use simulator ground truth from Physion++ with ThreeDWorld/Unity3D and Phys4D-style Isaac Sim scenes [37, 13, 26, 31]. For simulated clips, object poses, velocities, contacts, and support states provide structured evidence for candidate APTs; LLM/VLM prompting only converts this evidence into the shared APT schema and training formats, rather than inferring labels directly from pixels. Human-labeled simulated clips validate this conversion, while mixed sources and renderer diversity reduce visual-pattern shortcuts.

Using this mixed-source APT data, we first test whether current VLMs can naturally recover causal transition chains. They cannot: under the same 16-frame input and APT-schema prompt, eight zero-shot VLMs achieve only 9–14% recall in the full 14-type setting, with missed transitions as the dominant error rather than timestamp jitter. This indicates that clip-level event competence does not imply transition-level physical understanding. Directly fine-tuning on APT chains improves recall, but reveals a second failure mode: the model becomes an APT-format specialist. Because APT targets are structured JSON-like chains, whereas ordinary video tasks use QA, multiple choice, or short descriptions, APT-only SFT overfits to the output schema and weakens prompt-conditioned event-level competence. The key challenge is therefore to make transition-level causality reusable across answer formats.

Table 1: **Comparison of evaluation units across physics and temporal video benchmarks.** Prior work typically evaluates clip-level QA, future outcomes, full-clip continuation, or motion-grounded subsegments. APT differs by making timestamped, mechanism-typed physical transitions the supervision and scoring unit, enabling ordered chain-level evaluation.

Data / Eval.	Evaluation unit		Physical / temporal structure			Provenance	
	Target	Granularity	Temporal Location	Physics Taxonomy	Ordered Chain	Label / Source	Scale (native)
CLEVRER [41]	Causal video QA	Event QA	✗	Collision events	Partial	Sim GT	20K vids / >300K QA
CLEVRER-Humans [29]	Causal event QA	Event / edge QA	✗	Free-form events	Graph	Human on sim	1,108 vids / 1,076 QA
Physion [5]	Physical prediction	Future outcome	Endpoint	Object dynamics	✗	Sim GT + human	1.2K examples / 8 scen.
Physion++ [37]	Physical prediction	Future outcome	Endpoint	Latent properties	✗	Sim GT + human	8K train / 768 test [†]
PhysBench [9]	Physical VLM QA	Entry QA	✗	Broad domains	✗	Unspecified [†]	10,002 entries
Physics-IQ [33]	Video continuation	Future continuation	Partial	Physical laws	✗	Real video GT	396 vids / 66 scen.
MASS-Bench [38]	Physics video QA	Entity motion	Partial	Physics categories	Partial	Human+AI/GC	4,350 vids / 8,361 QA
TemporalBench [7]	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	27.3K APTs / 1,246 trials

We address this with APT-Tune, a parameter-efficient SFT recipe that treats APT enumeration as an atomic causal-representation pretext task. APT-Tune uses format-conditional co-training over APT, multiple-choice, and description formats, so the same video representation supports both local transition prediction and ordinary event-level answering. Non-APT formats provide event-level rehearsal, while image-pad-aware masking focuses supervised loss on the structured answer rather than multimodal padding artifacts. To encourage physical causality rather than label memorization, mechanism-conditioned APT prompts guide domain-to-type decoding, e.g., support loss under gravity to `free_fall_onset`, gap closure to `contact_init`, post-contact velocity reversal to `elastic_rebound`, and dissipated sliding to `sliding_arrest`. With only 11 M LoRA parameters on Qwen3-VL-2B, APT-Tune raises APT recall from 8.0% to 45.6%, transfers to PhysBench without training on it, and improves Something-Something v2 multiple-choice accuracy by 15 pp over the base model. These results show that APTs expose missing transition-level physics and provide format-robust causal supervision for broader video understanding.

Contributions. Our contributions are threefold:

- **We propose APTs as a human-inspired causal transition representation.** Atomic Physical Transitions represent videos as ordered chains of visually grounded physical state changes, exposing both when a regime changes and why it changes.
- **We construct mixed-source APT data for diagnosis and supervision.** Our data combines human annotations, simulator ground truth, and diverse renderers, revealing that current VLMs miss most transition-level physics even when clip-level events are recognizable.
- **We introduce APT-Tune for format-robust causal fine-tuning.** APT-Tune learns APTs as causal representations rather than a JSON output style, addressing format mismatch, event-level forgetting, and label memorization.

2 Related Work

Physics-oriented video and multimodal reasoning. Physics-oriented benchmarks have advanced causal video QA, physical prediction, latent-property inference, VLM evaluation, and law- or motion-aware video assessment [40, 4, 36, 11, 32, 44, 39, 27]. Their evaluation units, however, are typically clip-level answers, endpoint outcomes, plausibility scores, grounded QA targets, or trajectories. APT is complementary: it shifts the unit from event outcomes to temporally localized, mechanism-typed state changes, making transitions such as support loss, contact initiation, rebound, sliding arrest, and settling explicit units for annotation, evaluation, and supervision.

Temporal structure and transition-level supervision. APT is also related to event segmentation, physics expertise, and temporal video understanding. Cognitive studies suggest that humans segment activity around meaningful boundaries and that physics experts organize problems by underlying principles [43, 10, 21]. Video datasets and segmentation methods also model temporal structure, but usually through semantic actions, interactions, or procedural steps [15, 6, 12, 22, 1]. APT changes the boundary semantics from actions to physical regime transitions. This motivates APT-Tune: APT enumeration is used as transition-level causal supervision, while format-conditional



Figure 2: Event-level comparison between standard event-level evaluation and APT.

co-training and parameter-efficient adaptation [25, 23, 17] make the learned structure reusable across QA, multiple-choice, and description formats.

3 Atomic Physical Transitions: From Event Labels to Causal Chains

3.1 Events as Outcomes, APTs as Causes

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*, not by object or material category; rigid objects, soft bodies, granular materials, and fluid-like regions can instantiate the same APT when the active mechanism and visible before/after state change are the same. Event labels tell what happened; APT chains explain why it happened.

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

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

where τ_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 ℓ_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 not only *when* a state changes, but also *why* it changes.

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 τ_i contains no other named APT that further explains the same transition $s_i^- \rightarrow s_i^+$. In this sense, APTs are not physically indivisible events, but the smallest causal units we ask models to localize, classify, and reuse across answer formats.

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

3.3 Problem Formulation: APT Chain Enumeration

Let $V = (I_1, \dots, I_T)$ be a video with duration T . Transition-level physical understanding requires recovering an ordered chain of APTs:

$$\mathbf{A}(V) = (a_1, \dots, a_K), \quad 0 \leq \tau_1 \leq \dots \leq \tau_K \leq T, \quad a_i^{\text{eval}} = (\tau_i, \ell_i), \quad (2)$$

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.

An APT chain is a causal representation, not 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 learning it as a shared causal representation supports both transition enumeration and ordinary event-level understanding. We therefore formulate APT prediction as mechanism-conditioned decoding: first identify the active physical domain, then decode the transition type.

3.4 Scope and Boundary Rules

Our scope is macroscopic, visually grounded physical dynamics across rigid, deformable, granular, and fluid-like scenes. APTs are process-centered: a label is assigned by the active mechanism and visible before/after state change, not by the entity’s material category. We include localizable transitions such as contact onset or loss, support loss, collision response, friction- or medium-mediated motion change, surface or medium transition, rotation onset or arrest, stability loss, and settling. We do not estimate continuous parameters or full physical state, and we exclude changes without visible evidence; APTs therefore expose causal transition structure without requiring complete state or material reconstruction.

Labels are assigned to the entity, material region, or interacting pair undergoing the transition. Multiple APTs may share a timestamp if they describe independent changes, and ambiguous cases use the label that best explains the active mechanism, preferring specific transitions over generic ones. Detailed disambiguation rules are provided in the supplementary material.

3.5 APT Taxonomy

Our taxonomy contains 14 recurring, visually grounded physical transitions grouped by active process rather than object or material type. Rigid objects, soft bodies, granular materials, and fluid-like regions can instantiate the same APT when they share the same causal mechanism and before/after state change.

Contact & collision captures interaction-boundary changes and their outcomes, including contact onset/loss, rebound, and capture. **Gravity & projectile** captures support loss, apex reversal, and landing in gravity-driven motion. **Friction, surface & medium** captures motion changes caused by surface or medium interaction, including rest-to-motion, motion-to-rest, and transitions across

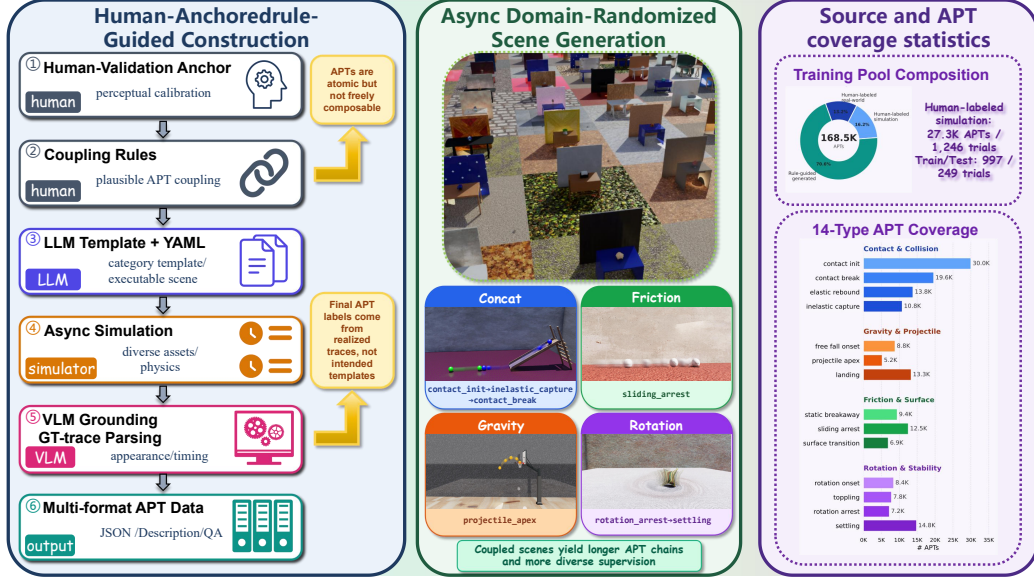


Figure 3: **Data collection pipeline for APT supervision.** Human validation anchors the APT semantics and coupling rules, while LLM/YAML templates are executed through asynchronous domain-randomized simulation to generate diverse coupled scenes. VLM grounding identifies visible objects, and simulator GT traces provide realized APT labels and timestamps. The resulting clips are exported as JSON, descriptions, and QA, yielding a broad training pool with full 14-type APT coverage.

173 surfaces or media. **Rotation & stability** captures angular-motion and stability changes, including
 174 rotation onset/arrest, toppling, and final equilibrium.

175 4 Rule-Guided Compositional APT Data Construction

176 4.1 Challenge and Overview

177 APT supervision is difficult to collect because it is finegrained, causal, and compositional. Unlike
 178 clip level labels, an APT chain must localize several short-lived state changes inside one video,
 179 such as support loss, contact initiation, rebound, frictional arrest, and settling, making direct human
 180 annotation slow and frame sensitive. Meanwhile, APTs are atomic but not freely composable:
 181 randomly combining transition types often yields physically incoherent scenes. We therefore propose
 182 a rule-guided coarse-to-fine pipeline to collect APT data.

183 4.2 Human-Labeled Validation Anchor

184 We use human annotation as the perceptual anchor for APT construction. Because APTs are
 185 visually grounded, simulator states alone are insufficient: transition boundaries must align with what
 186 humans can perceive in video. We annotate controlled physical videos from CLEVRER [41] and
 187 Physion++ [5, 37], where annotators mark visible state changes, assign one of the 14 APT types,
 188 and record timestamps. This yields 1,246 human-labeled trials with 27,303 APTs: 500 CLEVRER
 189 trials with 6,316 APTs covering 7 types, and 746 Physion++ trials with 20,987 APTs covering the
 190 full taxonomy. These labels are not our scaling mechanism; they calibrate templates, prompts, and
 191 GT-trace parsing so that generated APT chains match human-visible physical transitions.

192 4.3 Rule-Guided Coarse-to-Fine Simulation

193 We scale APT data with a rule-guided coarse2fine simulation pipeline that uses each component for
 194 what it is best suited for: humans provide physical common sense, LLMs expand and parse structured
 195 templates, coding agents make templates executable, simulators provide precise state traces, and

196 VLMs ground the rendered appearance. The pipeline starts from *human-defined coupling rules* rather
197 than arbitrary APT sequences, because APTs are atomic but not freely composable. For example,
198 gravity naturally couples with falling, landing, contact, rebound, and settling; friction couples with
199 sliding and arrest; and stability loss couples with rotation, toppling, landing, and settling. These rules
200 define a physically plausible coupling space.

201 Given a coupling rule, an LLM proposes several category-level scene templates under simulator
202 asset constraints, using supported classes such as dynamic rigid bodies, static surfaces, articulated
203 objects, garments, fluids, and sand. Each template specifies a causal scene skeleton which includes
204 the participating object categories, expected interactions, and coupled APT domains. For instance,
205 a friction-contact-garment rule may produce a template where a rigid object slides on a table and
206 stops due to friction, while a garment later falls onto it and settles. A coding agent then converts
207 the template into executable YAML, filling object slots, candidate assets, initial poses, velocities,
208 forces, material ranges, friction/restitution ranges, cameras, and success checks. We run these YAML
209 scenes with asynchronous parallel simulation and domain randomization, which instantiates concrete
210 objects, colors, materials, physical parameters, and initial conditions while preserving the intended
211 causal coupling.

212 The rendered video and simulator state traces are then refined into instance-specific APT data.
213 Because domain randomization chooses concrete assets only at simulation time, the original template
214 may not know whether the moving rigid object is a blue cube, a metal sphere, or a yellow cylinder.
215 We therefore use a VLM after rendering to ground the abstract template into the actual visible objects,
216 colors, materials, and relations. Precise APT stages are not inferred from pixels: an LLM-assisted
217 parser reads the simulator GT traces, including pose, velocity, angular velocity, contact pairs, support
218 relations, and object identity, and converts state changes into realized APT instances. For example,
219 support loss followed by downward motion yields `free_fall_onset`, a new contact pair yields
220 `contact_init`, post-contact relative velocity reversal yields `elastic_rebound`, and sustained low
221 motion yields `settling`. The intended template is used only as a prior; the final APT chain is
222 determined by the realized trace, allowing us to filter, relabel, or keep additional transitions caused by
223 randomized dynamics.

224 4.4 Validation, Formats, and Evaluation

225 We validate the automatic coarse-to-fine pipeline against the human-labeled anchor. The sampled
226 coupling rule is only an intended causal prior; final APT labels are determined from realized simulator
227 traces. We filter failed executions, relabel unintended outcomes, and use human comparison to
228 refine coupling rules, YAML success checks, grounding prompts, parsing rules, and motion/contact
229 thresholds. After calibration, automatic APT chains match human annotations closely enough to
230 scale beyond direct manual labeling.

231 Each accepted clip is exported in aligned formats: APT JSON with timestamps, grounded object
232 descriptions, causal transition descriptions, and event-level QA or multiple-choice examples. This
233 prevents APT supervision from collapsing into a single JSON output style and supports mixed-format
234 APT-Tune training.

235 Evaluation uses a frozen protocol: 16 uniformly sampled frames, one shared APT-schema prompt,
236 tolerant JSON parsing, and one-to-one matching by type equality within ± 200 ms. We report recall
237 with per-type breakdowns due to label imbalance. The human-labeled split is stratified by source and
238 scenario family, with 997 train trials (~ 21.8 K APTs) and 249 test trials (~ 5.5 K APTs).

239 5 APT-Tune: Learning from APT Chains

240 The benchmark establishes that transition-level physics can be labeled, generated, and evaluated.
241 This section describes how we turn human-labeled APT chains and generated APT-style chains into a
242 training method for VLMs. APT-Tune is built around two design choices: an atomic-representation
243 loss that trains the model to emit the timed transition chain, and format-conditional co-training that
244 keeps the model useful beyond the APT JSON format. Section 5.1 explains why both choices are
245 needed. The concrete LoRA instantiation and ablations are reported in Section 6.

Table 3: **The APT-Tune co-training mix.** Three formats co-trained on the same encoder. The APT format teaches the timed-event chain; MCQ and describe formats keep the model fluent in two answer schemas common in general video understanding.

Format	Prompt	Target	Source	N
apt	14-type APT schema	JSON list of timed events	APT-BENCH train + Isaac Lab	997 + generated
mcq	4-way action MCQ	Letter (A/B/C/D)	SSv2 [16]	1,000
describe	“Describe in one sentence”	Action label	SSv2	1,000

5.1 Two Principles

We treat APT enumeration as a *pretext task* for video VLMs: a structured target whose value lies not only in predicting the labels themselves, but also in the internal video representations the model must learn in order to emit a correct chain. Two principles guide the design of APT-Tune.

Principle 1: Atomic-Representation Loss with Image-Pad-Aware Masking. The APT chain is a structured target where every emitted token has a clear semantic role, such as a transition type, a timestamp, or syntactic scaffolding, and every video frame contributes to a small, well-defined set of timed events. Supervised fine-tuning over this target is therefore different from caption-style SFT: the loss signal is dense over physics structure rather than surface form, and semantically equivalent descriptions map to the same chain. For this principle to operate, the loss must flow only over the structured target. This requirement yields an image-pad-aware masking rule. In multimodal SFT, the processor expands an `<image>` placeholder into a grid of `<|image_pad|>` tokens; a text-only prefix length silently leaves ~ 500 pad tokens unmasked, turning part of the objective into pad-token reconstruction. Thus the prompt-prefix length must be obtained from the multimodal processor with the actual image inputs. Concretely, the standard recipe

$$L_{\text{text}} = \text{len}(\text{tokenizer}(\text{prompt}))$$

is replaced with the processor-aware variant

$$L_{\text{mm}} = \text{len}(\text{processor}(\text{prompt}, \text{images}).\text{input_ids}).$$

In our setting, this derivation of Principle 1 reduces final cross-entropy from 1.92 to 0.07, a $27\times$ reduction, and unlocks a 22 pp gain in APT-test recall on the same data and base model. Appendix ?? gives the implementation details.

Principle 2: Format-Conditional Co-Training. A naive specialist trained only on the APT format collapses the conditional output posterior $p(\text{format} \mid \text{prompt})$ onto APT-JSON, sacrificing competence on other tasks. We term this *specialist collapse* and find it empirically: a single-format APT specialist loses 4 pp on SSv2 MCQ relative to the un-fine-tuned base. To combat this, APT-Tune co-trains over a curated set of (prompt-template, target-format) pairs that share a single encoder, restoring prompt-conditional output:

- **APT format.** Prompt: 14-type APT schema. Target: JSON list of timed events.
- **MCQ format.** Prompt: 4-way action question. Target: a single letter (A/B/C/D).
- **Describe format.** Prompt: “Describe the action in one sentence.” Target: an action label.

The three formats share a single chat template; a record-level `format_type` field controls dispatch. The mix is small (Table 3) but format-diverse, and the MCQ/describe records are drawn from Something-Something v2 [16], an action-classification corpus with no overlap with our zero-shot evaluation splits. Importantly, the MCQ/describe records are not APT-label augmentation; their role is to maintain the format-conditional posterior while the APT objective trains atomic video structure.

Why this design matters. The method is not simply fine-tuning on another caption dataset. First, the target is a time-indexed chain of physical state transitions, so the model must learn to bind local visual evidence to a typed physical change. Second, the training objective must be aligned with the multimodal input representation; otherwise, the model spends capacity reconstructing padding artifacts instead of learning the APT chain. Third, the model must preserve ordinary instruction-following behavior while learning a specialized temporal schema. APT-Tune is, to our knowledge,

Table 4: **Zero-shot APT-BENCH results (8 VLMs)**. “CLEVRER subset” covers 7 APT types because gravitational and toppling APTs do not arise on a flat tabletop; “Physion++ portion” covers all 14. Recall and type accuracy are matched at ± 200 ms; ΔT is median timing error in ms over matched events. Best per column in bold.

Model	CLEVRER (7-type)			Physion++ (14-type)		
	Recall	Type-acc	ΔT	Recall	Type-acc	ΔT
GPT-4.1	.29	.43	90	.13	.15	95
Gemini 2.5 Flash	.27	.39	100	.12	.14	110
Qwen3-VL-2B	.26	.31	95	.10	.12	100
Qwen3-VL-4B	.31	.35	90	.11	.13	95
Qwen3-VL-8B	.35	.39	85	.12	.14	90
InternVL3.5-2B	.33	.35	80	.14	.16	67
InternVL3.5-4B	.36	.40	75	.13	.15	75
InternVL3.5-8B	.39	.42	80	.13	.15	80

the first video VLM SFT recipe that makes these three constraints explicit for transition-level physical understanding.

6 Experiments

We conduct experiments in four parts. First, we instantiate APT-Tune as a concrete LoRA recipe on Qwen3-VL-2B (Section 6.1). Second, we evaluate eight VLMs zero-shot to test whether current models already localize APT chains (Section 6.2). Third, we compare the un-fine-tuned base model with APT-Tune variants to validate the two design principles (Section 6.3). Finally, we analyze the dominant failure modes and per-type recovery patterns (Section 6.4). All numbers are on the APT-BENCH test split (249 trials, $\sim 5,500$ APTs across 14 types) under the protocol of Section ??: 16 frames, single APT-schema prompt, JSON output, and ± 200 ms matching tolerance. Transfer is measured on PhysBench [9] (100 real-world clips), Something-Something v2 MCQ [16] (200 clips, 4-way), and VideoPhy2 [2] (86 clips). Full evaluation details are in Appendix ??.

6.1 Training Recipe

We instantiate APT-Tune on Qwen3-VL-2B-Instruct [35]. We freeze the vision encoder and merger, and apply LoRA [18] ($r=16$, $\alpha=32$) only to the language-model attention and MLP projections. The trainable parameter count is ~ 11 M, roughly 0.5% of the full model. Training uses bf16 mixed precision, AdamW at $\text{lr } 1 \times 10^{-4}$ with a cosine schedule and 30 warm-up steps, and runs in under a day on a single A100. The APT-format training records are drawn from the human-labeled APT-BENCH train split and the Isaac Lab APT-style pool, while the MCQ and describe records are drawn from Something-Something v2.

Two trained variants. We train two APT-Tune variants on top of the same base and the same Principle-1 loss mask. APT-Tune-S (specialist) uses only the APT format. APT-Tune-G (generalist) uses the full multi-format mix of Table 3. This direct comparison isolates the contribution of Principle 2. A pre-Principle-1 variant **v1** (single-format APT with the text-only loss mask) provides a third reference point that isolates the contribution of Principle 1. We use APT-Tune-S and APT-Tune-G as the released specialist and generalist models, respectively, and report both because they form a Pareto front: APT-Tune-S dominates on APT detection, while APT-Tune-G dominates on general video understanding.

6.2 Zero-Shot Diagnostic

We evaluate eight VLMs zero-shot: GPT-4.1 [34], Gemini 2.5 Flash [14], Qwen3-VL-{2 B, 4 B, 8 B} [35], and InternVL3.5-{2 B, 4 B, 8 B} [8].

Table 4 reveals a sharp granularity gap. Recall plateaus at 25 to 40% on the 7-type CLEVRER subset and drops to 9 to 14% on the full 14-type portion; the best system (InternVL3.5-2B) reaches only 14.0% recall. Importantly, median timing error among *matched* events is 67 to 120 ms across every

Table 5: **Before and after APT-Tune on Qwen3-VL-2B.** Final loss is the cross-entropy at the end of training. The two principles of Section 5 are validated independently: removing Principle 1 (image-pad-aware mask) inflates the loss 27 $\times$ and costs 22 pp APT recall; removing Principle 2 (multi-format) regains 7 pp APT recall but loses 4 pp on SSv2 MCQ. Best per column in bold.

Model	Mask	Format	Final loss	APT recall	PhysBench	SSv2 MCQ
Baseline Qwen3-VL-2B	none	none	none	8.0	5.0	72.5
v1 (no Principle 1)	text-only	APT	1.92	23.7	none	50.0
APT-Tune-S (specialist)	mm-aware	APT	0.07	45.6	55.0	68.5
APT-Tune-G (generalist)	mm-aware	multi	0.13	38.1	52.0	87.5

model, well within the ± 200 ms matching tolerance, so the recall gap is dominated by missed detection, not jitter. Two APT types are at **0% across every model on Physion++**: `projectile_apex` and `inelastic_capture`; `surface_transition` is near 0%. Notably, scaling the open-source backbones from 2 B to 8 B yields only ~ 2 to 6 pp on the 7-type subset and no improvement on the 14-type portion, suggesting that the bottleneck is not capacity but representation.

6.3 APT-Tune Results

Table 5 reports the headline numbers. We first observe that the un-fine-tuned 2 B baseline reaches only 8.0% recall, below the best zero-shot 8 B systems, confirming that scale alone does not solve APT detection. With APT-Tune-S, recall rises to **45.6%**, type accuracy to **54.3%**, and median timing error halves from 74 to 33 ms; on PhysBench, the same model lifts hit rate from 5.0 to **55.0%**, a 50 pp jump on a benchmark we did not train on. On VideoPhy2, recall on physics violations rises from 5.9 to 47.1%. However, APT-Tune-S loses 4 pp on SSv2 MCQ relative to baseline (72.5 to 68.5), the empirical signature of specialist collapse.

APT-Tune-G resolves this trade-off. With the multi-format mix (Principle 2), APT recall settles at 38.1% (a 7 pp give-back relative to the specialist), PhysBench transfer is essentially preserved at 52.0%, and SSv2 MCQ rises from 72.5 to **87.5%**, a +15 pp gain over the un-fine-tuned base. Notably, this improvement appears while the model still preserves strong APT detection, consistent with our framing of APT enumeration as a pretext task when coupled with format-conditional co-training (Section 5.1). The two trained variants form a Pareto front: APT-Tune-S dominates on APT detection; APT-Tune-G dominates on general video understanding.

Effect of Principle 1. Comparing v1 (text-only mask) with APT-Tune-S (mm-aware mask), the loss collapses from 1.92 to 0.07 and APT recall jumps by 22 pp on the same data and base. The 1.92 plateau corresponds almost exactly to the model’s loss when reconstructing `<|image_pad|>` on the unmasked pad region, supporting the diagnosis in Section 5.1.

Effect of Principle 2. Comparing APT-Tune-S with APT-Tune-G, the multi-format mix preserves APT detection competence (38.1 vs. 45.6 recall) while restoring and exceeding general video competence (87.5 vs. 68.5 SSv2 MCQ). Importantly, this comparison isolates the practical role of Principle 2: APT-Tune-G gives back 7 pp APT recall relative to the specialist but gains 19 pp SSv2 MCQ, preventing the format collapse that single-format SFT exhibits.

6.4 Failure Analysis

Universal zeros and slice gradients. Per-type recall on Physion++ (Appendix ??) confirms that `projectile_apex` and `inelastic_capture` are at 0% recall for every zero-shot VLM, with `surface_transition` near 0%. APT-Tune recovers them to 12 to 25% recall, small in absolute terms but the first non-zero numbers in our setting. Across domains, contact APTs are most reliably detected (zero shot ~ 25 to 35% on the 7-type subset), gravity APTs intermediate, and friction/rotation hardest; APT-Tune preserves this ordering, suggesting it tracks visual saliency rather than training-set prevalence.

Missed detection dominates over jitter. Across all eight VLMs and both portions of APT-BENCH, median timing error of matched events stays between 67 and 120 ms. The recall gap is

therefore attributable to APTs the model fails to propose at all, a sensory rather than localization bottleneck. APT-Tune closes this gap by dense supervision over the structured APT target rather than by improving temporal precision: on matched events, APT-Tune-S reaches 33 ms median timing error, only marginally better than the 67 to 120 ms baselines.

Statistical reliability. All pairwise comparisons between the un-fine-tuned baseline and APT-Tune variants are significant under a paired bootstrap test on the 249 test trials ($p < 0.001$, $n=1,000$ resamples). Appendix ?? reports sensitivity to the temporal matching tolerance.

7 Discussion and Limitations

What APT-Bench is. APT-BENCH is a transition-level benchmark and data construction framework. The human-labeled component measures whether a model can enumerate APT chains in existing CLEVRER and Physion++ videos. The Isaac Lab component uses the same APT labels as scene construction targets, allowing us to efficiently generate APT-style clips whose physical parameters can be customized. This combination is important because existing videos provide realistic multi-step chains, while simulation lets us target rare or controlled transitions without waiting for them to appear naturally.

Limitation: current physics coverage. The current APT taxonomy mainly covers rigid-body and force-mediated transitions: contact, collision, gravity, friction, and rotation. This scope is deliberate, because these transitions are visually grounded, frame-localized, and easy to instantiate in controlled simulation. However, it does not cover all forms of physical change. Fluid motion, deformable-body dynamics, thermal processes, material fracture, granular media, electromagnetism, and long-horizon causal interactions are outside the present data. Future work will expand the taxonomy and generation pipeline to cover these broader APT families.

Limitation: annotation and simulation gap. Human labels provide semantic validity for real rendered videos, while Isaac Lab provides efficient controllability. The two sources are complementary but not identical. Some simulated clips may simplify visual texture, occlusion, or camera motion, while some human-labeled clips contain ambiguous boundaries that are hard to reproduce exactly in simulation. Future releases should report the two sources separately and study how much generated APT-style data improves transfer to real videos.

Future directions. APT labels can serve as both evaluation units and generation specifications. This suggests a scalable loop: define new APT labels, generate targeted clips in simulation, human-check boundary validity, and use the resulting examples to improve VLM training. We view this label-first data construction process as the most direct path toward broader physics coverage.

8 Conclusion

We introduce Atomic Physics Transitions and use them to construct APT-BENCH, a human-labeled benchmark of 1,246 CLEVRER and Physion++ trials and 27,303 timed APT events with full 14-type coverage. We further use the same APT labels as Isaac Lab simulation targets, enabling efficient and customizable construction of APT-style data for rare or controlled physical transitions. APT-Tune turns these APT chains into a parameter-efficient SFT recipe built on two principles: an atomic-representation loss with image-pad-aware masking, and format-conditional co-training that prevents specialist collapse. Eight zero-shot VLMs reach only 14% recall on the full taxonomy with failures dominated by missed detection. APT-Tune-S/APT-Tune-G on a 2 B base with 11 M LoRA parameters deliver +38 pp APT recall, +47 pp PhysBench transfer, and +15 pp SSv2 MCQ over the un-fine-tuned base, turning APT-Bench from a diagnostic into a training target that improves general video understanding. Current coverage focuses on rigid-body and force-mediated APTs; future work will expand the taxonomy to broader physical processes. Data, annotations, simulation construction protocol, evaluation code, and recipe are released.

References

- [1] Yazan Abu Farha and Juergen Gall. MS-TCN: Multi-stage temporal convolutional network for action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3575–3584, 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/html/Abu_Farha_MS-TCN_Multi-Stage_Temporal_Convolutional_Network_for_Action_Segmentation_CVPR_2019_paper.html.
- [2] Hritik Bansal et al. VideoPhy-2: Challenging physical commonsense for video generation. *arXiv preprint arXiv:2503.06800*, 2025.
- [3] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems*, volume 29, 2016. URL <https://proceedings.neurips.cc/paper/2016/hash/3147da8ab4a0437c15ef51a5cc7f2dc4-Abstract.html>.
- [4] Daniel M. Bear, Elias Wang, Damian Mrowca, Felix J. Binder, Hsiao-Yu Fish Tung, R. T. Pramod, Cameron Holdaway, Sirui Tao, Kevin A. Smith, Fan-Yun Sun, Fei-Fei Li, Nancy Kanwisher, Joshua B. Tenenbaum, Daniel L. K. Yamins, and Judith E. Fan. Physion: Evaluating physical prediction from vision in humans and machines. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL <https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/d09bf41544a3365a46c9077ebb5e35c3-Abstract-round1.html>.
- [5] Daniel M. Bear et al. Physion: Evaluating physical prediction from vision in humans and machines. In *NeurIPS Datasets and Benchmarks*, 2021.
- [6] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Nieves. ActivityNet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–970, June 2015. doi: 10.1109/CVPR.2015.7298698. URL https://openaccess.thecvf.com/content_cvpr_2015/html/Heilbron_ActivityNet_A_Large-Scale_2015_CVPR_paper.html.
- [7] Mu Cai, Reuben Tan, Jianrui Zhang, Bocheng Zou, Kai Zhang, Feng Yao, Fangrui Zhu, Jing Gu, Yiwu Zhong, Yuzhang Shang, et al. Temporalbench: Benchmarking fine-grained temporal understanding for multimodal video models. *arXiv preprint arXiv:2410.10818*, 2024.
- [8] Zhe Chen et al. InternVL3.5: Advancing open-source multimodal understanding. *arXiv preprint*, 2025.
- [9] Wei Cheng et al. PhysBench: Benchmarking and enhancing vision-language models for physical world understanding. In *ICLR*, 2025.
- [10] Michelene T. H. Chi, Paul J. Feltovich, and Robert Glaser. Categorization and representation of physics problems by experts and novices. *Cognitive Science*, 5(2):121–152, 1981. doi: 10.1207/s15516709cog0502_2.
- [11] 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. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/f38cb4cf9a5eaa92b3cfa481832719c6-Abstract-Conference.html.
- [12] 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. In *Proceedings of the European Conference on Computer Vision*, pages 720–736, September 2018. URL https://openaccess.thecvf.com/content_ECCV_2018/html/Dima_Damen_Scaling_Egocentric_Vision_ECCV_2018_paper.html.

- [13] Chuang Gan, Jeremy Schwartz, Seth Alter, Damian Mrowca, Martin Schrimpf, James Traer, Julian De Freitas, Jonas Kubilius, Abhishek Bhandwalder, Nick Haber, Megumi Sano, Kuno Kim, Elias Wang, Michael Lingelbach, Aidan Curtis, Kevin Feigels, Daniel M. Bear, Dan Gutfreund, David Cox, Antonio Torralba, James J. DiCarlo, Joshua B. Tenenbaum, Josh H. McDermott, and Daniel L. K. Yamins. Threedworld: A platform for interactive multi-modal physical simulation. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.
- [14] Google. Gemini 2.5 technical report. *Technical Report*, 2025.
- [15] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thureau, Ingo Bax, and Roland Memisevic. The “Something Something” video database for learning and evaluating visual common sense. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5842–5850, October 2017. doi: 10.1109/ICCV.2017.622. URL https://openaccess.thecvf.com/content_iccv_2017/html/Goyal_The_Something_Something_ICCV_2017_paper.html.
- [16] Raghav Goyal et al. The “something something” video database for learning and evaluating visual common sense. In *ICCV*, 2017.
- [17] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=nZeVKeeFYf9>.
- [18] Edward J. Hu et al. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.
- [19] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Fei-Fei Li, 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*, pages 2901–2910, July 2017. URL https://openaccess.thecvf.com/content_cvpr_2017/html/Johnson_CLEVR_A_Diagnostic_CVPR_2017_paper.html.
- [20] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2688–2697. PMLR, 2018. URL <https://proceedings.mlr.press/v80/kipf18a.html>.
- [21] Jill H. Larkin, John McDermott, Dorothea P. Simon, and Herbert A. Simon. Expert and novice performance in solving physics problems. *Science*, 208(4450):1335–1342, 1980. doi: 10.1126/science.208.4450.1335.
- [22] Colin Lea, Michael D. Flynn, René Vidal, Austin Reiter, and Gregory D. Hager. Temporal convolutional networks for action segmentation and detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 156–165, 2017. URL https://openaccess.thecvf.com/content_cvpr_2017/html/Lea_Temporal_Convolutional_Networks_CVPR_2017_paper.html.
- [23] Bin Lin, Yang Ye, Bin Zhu, Jiayi Cui, Munan Ning, Peng Jin, and Li Yuan. Video-LLaVA: Learning united visual representation by alignment before projection. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 5971–5984, Miami, Florida, USA, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.342. URL <https://aclanthology.org/2024.emnlp-main.342/>.
- [24] Samuel J. Ling, Jeff Sanny, and William Moebs. *University Physics Volume 1*. OpenStax, Rice University, 2016. URL <https://openstax.org/details/books/university-physics-volume-1>.
- [25] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://papers.neurips.cc/paper_files/paper/2023/hash/6dcf277ea32ce3288914faf369fe6de0-Abstract-Conference.html.

- [26] Haoran Lu, Shang Wu, Jianshu Zhang, Maojiang Su, Guo Ye, Chenwei Xu, Lie Lu, Pranav Maneriker, Fan Du, Manling Li, Zhaoran Wang, and Han Liu. Phys4d: Fine-grained physics-consistent 4d modeling from video diffusion. *arXiv preprint arXiv:2603.03485*, 2026.
- [27] Chak-Wing Mak, Guanyu Zhu, Boyi Zhang, Hongji Li, Xiaowei Chi, Kevin Zhang, Yichen Wu, Yangfan He, Chun-Kai Fan, Wentao Lu, Kuangzhi Ge, Xinyu Fang, Hongyang He, Kuan Lu, Tianxiang Xu, Li Zhang, Yongxin Ni, Youhua Li, and Shanghang Zhang. PhysicsMind: Sim and real mechanics benchmarking for physical reasoning and prediction in foundational vlms and world models. *arXiv preprint arXiv:2601.16007*, 2026. doi: 10.48550/arXiv.2601.16007. URL <https://arxiv.org/abs/2601.16007>.
- [28] 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, Hongyang He, Kuan Lu, Tianxiang Xu, Li Zhang, Yongxin 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>.
- [29] Jiayuan 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>.
- [30] Kexin Meng et al. MORPHEUS: Benchmarking physical reasoning of video generative models with real physical experiments. *arXiv preprint arXiv:2504.02918*, 2025.
- [31] Mayank Mittal et al. Isaac Lab: A GPU-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025.
- [32] Saman Motamed, Laura Culp, Kevin Swersky, Priyank Jaini, and Robert Geirhos. Do generative video models understand physical principles? In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 948–958, March 2026. URL https://openaccess.thecvf.com/content/WACV2026/html/Motamed_Do_Generative_Video_Models_Understand_Physical_Principles_WACV_2026_paper.html.
- [33] Saman Motamed et al. Physics-IQ: Benchmarking physical understanding in video models. *arXiv preprint arXiv:2501.09038*, 2025.
- [34] OpenAI. GPT-4o technical report. *Technical Report*, 2024.
- [35] Qwen Team. Qwen3-VL: Multimodal understanding at scale. *Technical Report*, 2025.
- [36] Hsiao-Yu Tung, Mingyu Ding, Zhenfang Chen, Daniel Bear, Chuang Gan, Joshua B. Tenenbaum, Daniel L. K. Yamins, Judith E. Fan, and Kevin A. Smith. Physion++: Evaluating physical scene understanding that requires online inference of different physical properties. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/d3e8011c912e651ab2a76e7935a1e464-Abstract-Datasets_and_Benchmarks.html.
- [37] Hsiao-Yu Tung et al. Physion++: Evaluating physical scene understanding that requires online inference of different physical properties. In *NeurIPS Datasets and Benchmarks*, 2023.
- [38] Xiao Wang et al. MASS: Motion-aware spatial-temporal grounding for physics reasoning and comprehension in vision-language models. *arXiv preprint arXiv:2511.18373*, 2025.
- [39] Xiyang Wu, Zongxia Li, Jihui Jin, Guangyao Shi, Gouthaman KV, 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 preprint arXiv:2511.18373*, 2025. doi: 10.48550/arXiv.2511.18373. URL <https://arxiv.org/abs/2511.18373>.
- [40] Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B. Tenenbaum. CLEVRER: CoLLision events for video representation and reasoning. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=HkxYzANYDB>.

- [41] Kexin Yi et al. CLEVRER: Collision events for video representation and reasoning. In *ICLR*, 2020.
- [42] Hugh D. Young and Roger A. Freedman. *University Physics with Modern Physics*. Pearson, 15 edition, 2020.
- [43] Jeffrey M. Zacks, Nicole K. Speer, Khena M. Swallow, Todd S. Braver, and Jeremy R. Reynolds. Event perception: A mind-brain perspective. *Psychological Bulletin*, 133(2):273–293, 2007. doi: 10.1037/0033-2909.133.2.273.
- [44] Chenyu Zhang, Daniil Cherniavskii, Antonios Tragoudaras, Antonios Vozikis, Thijmen Nijdam, Derck W. E. Prinzhorn, Mark Bodracska, Nicu Sebe, Andrii Zadaianchuk, and Efstratios Gavves. Morpheus: Benchmarking physical reasoning of video generative models with real physical experiments. *arXiv preprint arXiv:2504.02918*, 2025. doi: 10.48550/arXiv.2504.02918. URL <https://arxiv.org/abs/2504.02918>.

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 [40]. 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 [4, 36]. PhysBench broadens the evaluation scope to VLMs and interleaved video-image-text physical reasoning [11]. More recent work evaluates whether generated or mixed-modality videos obey physical principles, including Physics-IQ, Morpheus, MASS/MASS-Bench, and PhysicsMind [32, 44, 39, 27]. 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 [43]. Second, studies of physics expertise show that experts tend to organize problems by underlying physical principles rather than surface-level objects or scene appearance [10, 21]. 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 [15, 6, 12]. Temporal action segmentation further assigns frame- or segment-level semantic action labels over long videos [22, 1]. 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.

Work	Primary unit	Setting	Temporal localization	Mechanism-typed transitions	Ordered-chain of
CLEVRER [40]	causal / counterfactual video QA	synthetic video	limited	no	no
Physion [4]	physical outcome prediction	synthetic video	endpoint-level	no	no
Physion++ [36]	latent-property physical prediction	synthetic video	endpoint-level	no	no
PhysBench [11]	VLM physical-world reasoning	mixed image/video/text	limited	no	no
Physics-IQ [32]	law-aware video-generation evaluation	real video	trajectory-level	no	no
Morpheus [44]	physical-law evaluation of generated video	real physical experiments	trajectory-level	no	no
MASS/MASS-Bench [39]	physics QA with motion and sub-segment grounding	real + AIGC video	yes	not explicit	no
PhysicsMind [27]	VQA and physics-aware generation	real + simulation	limited	no	no
Something-Anything [15]	fine-grained human-object action label	real video	clip-level	no	no
ActivityNet [6]	activity class and temporal segment	real video	yes	no	no
EPIC-KITCHENS [12]	egocentric action/object segment	real video	yes	no	no
APT framework	mechanism-typed APT chain	real + simulation	yes	yes	yes

Table 6: Representative comparison along the axis most relevant to our framework. The table is not intended as a broad ranking of physical reasoning benchmarks; it isolates whether the evaluated or supervised unit is an ordered chain of temporally localized, mechanism-typed physical transitions.

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 [3, 20]. Synthetic physical reasoning benchmarks such as CLEVRER, Physion, and Physion++ similarly benefit from simulator access and controlled physical state information [40, 4, 36]. APTs are complementary to these approaches. Rather than replacing full state rollout, object-centric prediction, or simulation, APTs provide a compact causal abstraction 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 [19, 40]. Physics-oriented benchmarks further test whether models can reason about counterfactual outcomes, physical plausibility, and law-consistent prediction or generation [11, 32, 44, 27]. 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 [25, 23]. Parameter-efficient methods such as LoRA make such adaptation practical for large models [17]. 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 causal 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 [6, 12]. CLEVRER, Physion, and Physion++ illustrate the value of controlled simulation and privileged physical state information for systematic physical reasoning evaluation [40, 4, 36]. 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 used to make APTs a causal representation rather than merely a fine-grained event vocabulary.

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 causal 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 causal process 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 causal 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

$$a_i = (\tau_i, d_i, \ell_i, s_i^- \rightarrow s_i^+, e_i), \quad (3)$$

where τ_i is the canonical transition time, d_i is the physical domain, ℓ_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. This representation should be interpreted as follows.

Canonical time. The timestamp τ_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 supports mechanism-conditioned decoding in APT-Tune: the model first identifies the active mechanism family and then decodes the transition type within that family.

Transition type. The type ℓ_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 $s_i^- \rightarrow s_i^+$ 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 causal transition from approach to retreat. APTs are therefore the smallest causal units that we ask models to localize, classify, and reuse across answer formats.

B.3 APT Chains and Output Serializations

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

$$\mathbf{A}(V) = (a_1, \dots, a_K), \quad 0 \leq \tau_1 \leq \dots \leq \tau_K \leq T, \quad (4)$$

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

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

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"},
```

```

716     {"time": 2.10, "type": "settling"}
717 ]
718
719 Description:
720 "The ball hits the table, bounces away, lands again, and settles."
721
722 QA:
723 Q: What happens after the ball contacts the table?
724 A: It rebounds and then separates from the table.
725
726 Multiple choice:
727 The ball (A) passes through the table, (B) rebounds after contact,
728 (C) remains floating, or (D) disappears. Answer: B.

```

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

732 B.4 Scope

733 Our scope is macroscopic, visually grounded rigid-body mechanics. We include physical state
734 changes that satisfy three criteria.

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

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

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

743 We do not attempt to estimate continuous physical parameters such as mass, force magnitude, friction
744 coefficient, restitution coefficient, full 3D pose, or exact contact forces. We also exclude non-rigid
745 phenomena outside the scope of our data, such as fluids, fracture, thermal change, melting, tearing, or
746 material deformation that is not visually represented as a rigid-body interaction. In this sense, APTs
747 form an intermediate representation between clip-level event recognition and full physical simulation.
748 They expose causal process structure without requiring complete state reconstruction.

749 B.5 Boundary and Disambiguation Rules

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

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

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

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

770 **Elastic rebound versus contact break.** `elastic_rebound` marks the visible reversal from ap-
771 proach to retreat after collision. `contact_break` marks the later loss of contact, when the objects
772 or the object and surface visibly separate. In a bounce, these may be close in time but they are not
773 the same transition: rebound is the change in motion direction caused by collision response, while
774 contact break is the change in contact state.

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

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

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

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

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

796 B.6 Taxonomy by Example

797 Table 8 expands the compact taxonomy in the main text with example annotation cues. The examples
798 are written as visual-causal cues: what changes in the video, and why that change is a distinct physical
799 transition.

800 B.7 Domain-Level Explanation

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

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

Table 7: **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. support formation vs. generic contact	Use <code>landing</code> when a falling object becomes supported; use <code>contact_init</code> for lateral or generic contact onset.
contact_init		
free_fall_onset	vs. gravity-driven fall vs. generic separation	Use <code>free_fall_onset</code> when support loss causes falling; use <code>contact_break</code> when separation is the main transition.
contact_break		
elastic_rebound	vs. velocity reversal vs. contact loss	Use <code>elastic_rebound</code> for approach-to-retreat reversal; use <code>contact_break</code> for visible separation.
contact_break		
inelastic_capture	collision capture vs. frictional stop	Use <code>inelastic_capture</code> when impact dissipates motion into rest; use <code>sliding_arrest</code> when sliding motion decays to rest.
vs. sliding_arrest		
sliding_arrest	vs. end of sliding vs. final equilibrium	Use <code>sliding_arrest</code> for translational sliding-to-rest; use <code>settling</code> for final stabilization after residual motion.
settling		
rotation_arrest	vs. angular stop vs. full stabilization	Use <code>rotation_arrest</code> when visible angular motion stops; use <code>settling</code> when all residual motion ends.
settling		
toppling	vs. loss of stability vs. generic angular onset	Use <code>toppling</code> when an object loses stability and falls over; use <code>rotation_onset</code> for generic start of spin or angular motion.
rotation_onset		
surface_transition	surface change vs. falling-to-supported	Use <code>surface_transition</code> when a supported object crosses surfaces; use <code>landing</code> when a falling object becomes supported.
vs. landing		

812 marks the transition from upward to downward vertical motion. `landing` marks the falling-to-
813 supported transition. An event-level label such as “falls” merges these phases, while the APT chain
814 records the causal structure of the trajectory.

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

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

826 B.8 Label-Level Annotation Notes

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

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

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

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

Table 8: **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 (<i>regime changes at contact boundaries</i>)			
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 (<i>regime changes under gravity</i>)			
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 (<i>friction-mediated regime changes</i>)			
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 (<i>rotational and stability regime changes</i>)			
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.

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

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

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

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

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

855 **surface_transition.** Mark the moment an object crosses from one supporting surface or surface
856 condition to another. Examples include ramp-to-floor, smooth-to-rough, or low-friction-to-high-

Table 9: **Example event labels and possible APT chains.** Event labels describe clip-level outcomes, while APT chains expose the causal transitions that make those outcomes unfold.

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

857 friction regions. The transition should be accompanied by a visible or expected change in motion
858 caused by the new surface condition.

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

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

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

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

871 B.9 APT Chains by Example

872 Table 9 illustrates how coarse event labels are expanded into APT chains. The purpose is not to
873 prescribe a unique chain for every possible video, but to show how event names become causal
874 transition sequences.

875 B.10 APT Labels as Simulation Targets

876 The APT definition serves two roles. First, it tells annotators what transition to mark in an existing
877 video. Second, it provides a label-first specification for simulation. Once an APT label or chain is
878 fixed, a simulator can be configured to instantiate that transition by controlling geometry, mass, initial
879 velocity, applied force, friction, restitution, support, and layout. For example, an **elastic_rebound**
880 scene can be produced by setting a rigid barrier and sufficient restitution; a **sliding_arrest** scene
881 can be produced by setting an initial sliding velocity and frictional surface; a **toppling** scene can be
882 produced by applying an off-center push to a tall object near its stability boundary.

883 This label-first route is important because APTs need not be discovered passively. Rare transitions
884 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.11 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:

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

This factorization reflects the causal 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. In the main method, this idea is implemented through mechanism-conditioned APT prompts and format-conditional co-training. The causal scaffold encourages the model to learn APTs as reusable physical representations rather than as a narrow JSON output schema.

B.12 Summary of the Supplementary Definition

APTs are defined to capture the causal 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 deliberately scoped to macroscopic rigid-body 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 causal fine-tuning.

C Supplementary Details: Rule-Guided Coarse-to-Fine APT Construction

This section provides implementation details for our rule-guided coarse-to-fine APT construction pipeline. 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` as the VLM for post-render semantic grounding. The central design principle is to use each component for the part of the pipeline where it is most reliable: humans provide physical common sense, LLMs expand and parse structured templates, coding agents make templates executable, simulators execute physics and record ground-truth state traces, and VLMs identify concrete visual objects after rendering. Thus, LLMs and VLMs are not treated as the source of physical truth. Final APT labels are determined from realized simulator traces and calibrated against human-visible transitions.

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 [41] and Physion++ [5, 37]: annotators watch each clip, mark visible physical state changes, assign one of the 14 APT types, and record transition timestamps. The human-labeled anchor contains 500 CLEVRER validation 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 in total. We use this anchor to calibrate coupling rules, prompts, YAML success checks, simulator-trace parsing rules, and motion/contact thresholds.

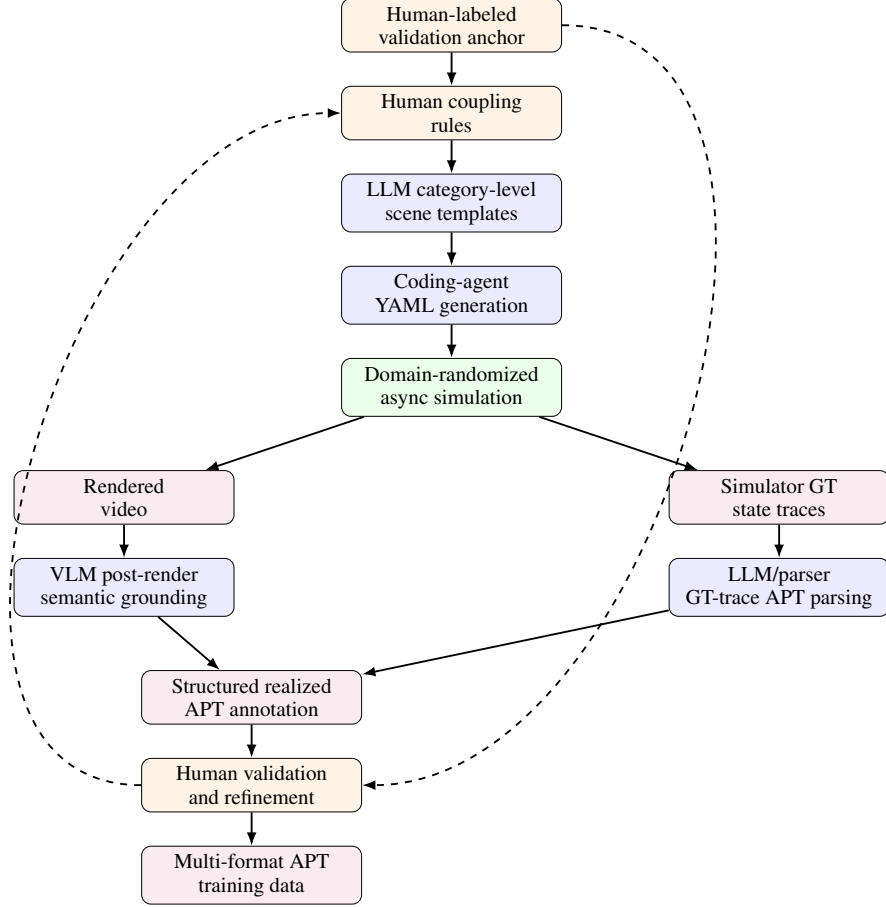


Figure 4: **Rule-guided coarse-to-fine APT construction.** Humans provide a perceptual anchor and admissible causal couplings. An LLM proposes category-level templates, a coding agent converts them to executable YAML, and asynchronous simulation executes them under domain randomization. After rendering, 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.

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

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

940 C.3 Step 1: Human Coupling Rules

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

Listing 1: Demo human-defined coupling rule.

```

945
946 Allowed domains:
947     friction, contact, gravity, settling
948
949 Allowed APT types:
950     static_breakaway
951     surface_transition
952     sliding_arrest
953     free_fall_onset
954     contact_init
955     landing
956     inelastic_capture
957     settling
958
959 Physical sense:
960     A dynamic rigid object can slide on a support surface, cross into a rougher region, and stop due to
961     friction.
962     A garment can become unsupported, fall under gravity, contact the rigid object or support surface, and
963     settle without rebound.
964     These transitions are physically compatible and can form a longer APT chain than a simple single-event
965     clip.
966
967 Simulator asset classes allowed:
968     dynamic_rigid, static_rigid, articulated, garment, fluid, sand

```

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

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

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

```

975
976 You are helping construct simulation scenes for Atomic Physical Transitions (APTs).
977 The goal is to create physically plausible scenes that contain multiple coupled APTs.
978
979 Supported simulator asset classes:
980     dynamic_rigid
981     static_rigid
982     articulated
983     garment
984     fluid
985     sand
986
987 Human-defined coupling rule:
988     Friction, contact, gravity, and settling may co-occur.
989     A dynamic rigid object may slide on a support surface and stop due to friction.
990     A garment may fall onto the rigid object or support surface and then settle.
991
992 Target APT domains:
993     friction, contact, gravity, settling
994
995 Instructions:
996     Propose 3 category-level scene templates.
997     For each template, output:
998         (1) scene name,
999         (2) object slots and their simulator asset classes,
1000         (3) natural-language causal scene skeleton,
1001         (4) expected APT types,
1002         (5) success checks,
1003         (6) likely failure cases.
1004
1005 Do not choose final object identity, color, or material.
1006 Do not output timestamps.

```

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

```

1008
1009 Template 1: rigid-slide-garment-cover
1010
1011 Object slots:
1012     support: static_rigid, e.g., table or platform
1013     mover: dynamic_rigid, e.g., cube, sphere, cylinder, mug, or apple
1014     cover: garment, e.g., pants, shorts, or cloth
1015

```

```

1016 Causal scene skeleton:
1017   A dynamic rigid object starts on a support surface, receives an initial horizontal velocity,
1018   slides across the surface, crosses into a higher-friction region, and stops.
1019   A garment is released above the rigid object, falls under gravity, contacts the rigid object
1020   or the support surface, and settles.
1021
1022 Expected APTs:
1023   static_breakaway
1024   surface_transition
1025   sliding_arrest
1026   free_fall_onset
1027   contact_init
1028   landing
1029   inelastic_capture
1030   settling
1031
1032 Success checks:
1033   The rigid object should move tangentially for at least 0.3 s and later remain below a velocity threshold.
1034
1035   The garment should have a clear downward motion phase, create a new contact pair, and remain
1036   approximately at rest.
1037
1038 Failure cases:
1039   The rigid object never moves.
1040   The garment misses the support area.
1041   The garment penetrates the table.
1042   The garment remains airborne at the end.
1043   The rigid object rebounds instead of stopping.
1044
1045 Template 2: rigid-bump-then-cloth-damping
1046
1047 Object slots:
1048   support: static_rigid
1049   mover: dynamic_rigid
1050   obstacle: static_rigid or dynamic_rigid
1051   cover: garment
1052
1053 Causal scene skeleton:
1054   A dynamic rigid object slides into a low obstacle, changes direction or starts rotating,
1055   slows on a rough surface, and is later covered by a falling garment that damps residual motion.
1056
1057 Expected APTs:
1058   static_breakaway
1059   contact_init
1060   rotation_onset
1061   sliding_arrest
1062   free_fall_onset
1063   landing
1064   settling
1065
1066 Template 3: falling-rigid-with-garment-capture
1067
1068 Object slots:
1069   support: static_rigid
1070   falling_object: dynamic_rigid
1071   cover: garment
1072
1073 Causal scene skeleton:
1074   A small dynamic rigid object falls from above a support surface, lands with low restitution,
1075   slides briefly, and is later covered by a falling garment.
1076
1077 Expected APTs:
1078   free_fall_onset
1079   landing
1080   contact_init
1081   inelastic_capture
1082   sliding_arrest
1083   settling

```

1085 C.5 Step 3: Coding-Agent YAML Generation

1086 A coding agent converts the selected category-level scene template into an executable YAML scene.
1087 This step makes the template simulatable and introduces diversity in initial conditions. The YAML
1088 specifies object slots, candidate assets, randomized positions, initial velocities, material ranges,
1089 friction and restitution ranges, camera and lighting settings, success checks, and trace logging. All
1090 asset paths below are anonymized.

Listing 4: Prompt for coding-agent YAML generation.

```

1091 Convert the following category-level APT template into an executable YAML scene.
1092
1093
1094 Template name:
1095     rigid-slide-garment-cover
1096
1097 Scene skeleton:
1098     A dynamic rigid object slides on a table, crosses into a rougher region,
1099     slows due to friction, and stops.
1100     A garment is released above the rigid object, falls onto it or the table,
1101     and settles.
1102
1103 Available anonymized asset roots:
1104     Tables:
1105         ~/Desk/...
1106     Dynamic rigid objects:
1107         ~/Object/Mug
1108         ~/Object/apple
1109         ~/Object/Sugar_Box
1110     Garments:
1111         ~/Object/Garment/Pants
1112     Materials:
1113         ~/Material/...
1114     Backgrounds:
1115         ~/Background/...
1116
1117 Requirements:
1118     Use domain randomization over asset identity, color, material, initial pose,
1119     initial velocity, friction, restitution, lighting, and camera.
1120     Record per-frame pose, linear velocity, angular velocity, contact pairs,
1121     support relation, object identity, material parameters, and surface region ID.
1122     Add success checks for sliding motion, surface transition, arrest, garment falling,
1123     garment contact, and final settling.
1124     Return YAML only.

```

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

```

1126 scene_name: rigid_slide_garment_cover
1127 description: >
1128     A dynamic rigid object slides on a table, enters a higher-friction region,
1129     stops, and is later covered by a falling garment.
1130
1131 background:
1132     intensity: [1000, 2000]
1133     texture:
1134         - "~/Background/champagne_castle_1_4k.hdr"
1135         - "~/Background/evening_road_01_4k.hdr"
1136         - "~/Background/kloofendal_48d_partly_cloudy_4k.hdr"
1137         - "~/Background/kloppenheim_02_4k.hdr"
1138         - "~/Background/mealie_road_4k.hdr"
1139         - "~/Background/moonlit_golf_4k.hdr"
1140         - "~/Background/noon_grass_4k.hdr"
1141         - "~/Background/qwantani_4k.hdr"
1142
1143 light:
1144     light_1:
1145         types: ["Rect", "Sphere", "Cylinder"]
1146         common:
1147             initial_pos_range: [-0.2, -0.2, 0.0, 0.2, 0.5, 0.0]
1148             initial_ori_range: [-30, -30, -30, 30, 30, 30]
1149             color_temperature_range: [3000, 9000]
1150             color_range: [[0.8, 0.8, 0.8], [1.2, 1.2, 1.2]]
1151         Rect:
1152             intensity_range: [2000, 5000]
1153             width_range: [1.0, 3.0]
1154             height_range: [1.0, 3.0]
1155         Sphere:
1156             intensity_range: [5000, 10000]
1157             radius_range: [0.1, 0.5]
1158         Cylinder:
1159             intensity_range: [4000, 8000]
1160             radius_range: [0.1, 0.3]
1161             length_range: [1.0, 2.0]
1162
1163 camera:
1164     random: true
1165     look_at: [0.0, 0.2, 0.75]
1166     pos_range: [-1.8, -2.2, 1.2, 1.8, -1.2, 2.2]
1167     focal_length_range: [24, 45]

```

```

1169
1170 objects:
1171   support_table:
1172     usd:
1173       - "~/Desk/Collected_appleseed_coffeetable/appleseed_coffeetable_inst_base.usd"
1174       - "~/Desk/Collected_cline/cline_inst_base.usd"
1175       - "~/Desk/Collected_oaktablesmall/oaktablesmall_inst_base.usd"
1176       - "~/Desk/Collected_willowbench/willowbench_inst_base.usd"
1177     random: true
1178     default_material: true
1179     num_per_env: 1
1180     semantic_label: "static_support_surface"
1181     support_table_1:
1182       pos: [0.0, 0.0, 0.0]
1183       ori: [0.0, 0.0, 90.0]
1184       visual:
1185         scale: [2.0, 2.0, 1.0]
1186         color:
1187           - [0.8, 0.7, 0.6]
1188           - [0.6, 0.5, 0.4]
1189           - [0.4, 0.3, 0.2]
1190           - [0.2, 0.1, 0.0]
1191       visual_material:
1192         mdl_folder: "~/Material/Base/Textiles"
1193     physics:
1194       type: "geometry"
1195       collision: true
1196       surface_regions:
1197         low_friction_region:
1198           x_range: [-0.70, -0.05]
1199           static_friction_range: [0.05, 0.20]
1200           dynamic_friction_range: [0.05, 0.20]
1201         high_friction_region:
1202           x_range: [-0.05, 0.70]
1203           static_friction_range: [0.80, 1.30]
1204           dynamic_friction_range: [0.70, 1.20]
1205
1206   mover:
1207     common:
1208       primitive_scale: [0.30, 0.30, 0.30]
1209       initial_pos_range: [-0.60, -0.05, 0.75, -0.45, 0.20, 0.82]
1210       initial_ori_range: [-10, -10, -20, 10, 10, 20]
1211     usd:
1212       - "~/Object/Mug"
1213       - "~/Object/apple"
1214       - "~/Object/Sugar_Box"
1215     random: true
1216     num_per_env: 1
1217     semantic_label: "dynamic_rigid_mover"
1218     mover_1:
1219       visual:
1220         scale: [1.0, 1.0, 1.0]
1221         visible: true
1222         color:
1223           - [1.0, 0.0, 0.0]
1224           - [0.0, 0.0, 1.0]
1225           - [1.0, 1.0, 0.0]
1226           - [0.0, 1.0, 1.0]
1227       visual_material:
1228         mdl_folder: "~/Material/Base/Textiles"
1229     physics:
1230       type: "dynamic"
1231       collision: true
1232       mass_range: [0.2, 0.8]
1233       ratio: 1.1
1234       physics_material:
1235         static_friction_range: [0.10, 0.35]
1236         dynamic_friction_range: [0.10, 0.35]
1237         restitution_range: [0.0, 0.10]
1238       linear_velocity_range:
1239         - [0.9, -0.05, 0.0]
1240         - [1.4, 0.05, 0.0]
1241       angular_velocity_range:
1242         - [0.0, 0.0, -0.5]
1243         - [0.0, 0.0, 0.5]
1244       track_contact_forces: true
1245       prepare_contact_sensor: true
1246
1247   garment_cover:
1248     common:
1249       primitive_scale: [0.50, 0.50, 0.50]

```

```

1250     initial_pos_range: [0.05, -0.10, 1.35, 0.35, 0.25, 1.70]
1251     initial_ori_range: [-15, -15, 60, 15, 15, 120]
1252     usd:
1253       - "~/Object/Garment/Pants"
1254     random: true
1255     num_per_env: 1
1256     semantic_label: "falling_garment"
1257     garment_cover_1:
1258       visual:
1259         scale: [1.0, 1.0, 1.0]
1260         visible: true
1261         color:
1262           - [1.0, 0.0, 0.0]
1263           - [0.0, 1.0, 0.0]
1264           - [0.0, 0.0, 1.0]
1265           - [1.0, 1.0, 0.0]
1266           - [1.0, 0.0, 1.0]
1267           - [0.0, 1.0, 1.0]
1268       visual_material:
1269         mdl_folder: "~/Material/Base/Textiles"
1270     physics:
1271       type: "garment"
1272       collision: true
1273       ratio: 1.1
1274       release_time_range: [0.55, 0.90]
1275       particle_system:
1276         particle_system_enabled: true
1277         enable_ccd: true
1278         solver_position_iteration_count: 16
1279         max_depenetration_velocity: null
1280         global_self_collision_enabled: true
1281         non_particle_collision_enabled: true
1282         contact_offset: 0.010
1283         rest_offset: 0.006
1284         particle_contact_offset: 0.010
1285         fluid_rest_offset: 0.006
1286         solid_rest_offset: 0.006
1287       particle_material:
1288         adhesion: 10
1289         adhesion_offset_scale: 0.0
1290         cohesion: 0.1
1291         particle_adhesion_scale: 3.0
1292         particle_friction_scale: 3.0
1293         drag: 0.0
1294         lift: 0.0
1295         friction: 10.0
1296         damping: 5.0
1297         gravity_scale: 1.0
1298       garment_config:
1299         particle_mass: 5e-10
1300         self_collision: true
1301         self_collision_filter: true
1302         stretch_stiffness: 1e12
1303         bend_stiffness: 1e3
1304         shear_stiffness: 1e3
1305         spring_damping: 5.0
1306
1307     trace_logging:
1308       fps: 60
1309       fields:
1310         - pose
1311         - linear_velocity
1312         - angular_velocity
1313         - contact_pairs
1314         - contact_forces
1315         - support_relation
1316         - object_identity
1317         - material_parameters
1318         - surface_region_id
1319
1320     success_checks:
1321       intended_domains:
1322         - friction
1323         - contact
1324         - gravity
1325         - settling
1326       required_conditions:
1327         mover_slides:
1328           object: "mover"
1329           min_horizontal_displacement: 0.25
1330           min_duration_sec: 0.25

```

```

1331     surface_transition:
1332       object: "mover"
1333       from_region: "low_friction_region"
1334       to_region: "high_friction_region"
1335     sliding_arrest:
1336       object: "mover"
1337       speed_below: 0.03
1338       hold_duration_sec: 0.20
1339     garment_falls:
1340       object: "garment_cover"
1341       min_downward_displacement: 0.30
1342     garment_contacts:
1343       object: "garment_cover"
1344       target_any_of: ["mover", "support_table"]
1345     final_settling:
1346       objects: ["mover", "garment_cover"]
1347       linear_speed_below: 0.03
1348       angular_speed_below: 0.10
1349       hold_duration_sec: 0.30
1350     failure_filters:
1351       - "garment misses table and falls out of view"
1352       - "mover never enters high-friction region"
1353       - "mover leaves support surface"
1354       - "simulation contains severe interpenetration"

```

1356 C.6 Step 4: Domain-Randomized Asynchronous Simulation

1357 We execute each YAML scene in Isaac Lab [31] with asynchronous parallel simulation and domain
1358 randomization. The same category-level template can produce many concrete videos because asset
1359 identity, color, material, physical parameters, initial pose, initial velocity, camera, and lighting are
1360 sampled at simulation time. For example, the abstract slot `dynamic_rigid_mover` may render as a
1361 blue cube, a red mug, a yellow cylinder, or a green apple; the `falling_garment` slot may render
1362 as red pants, blue pants, or a cloth-like garment. This design increases both visual diversity and
1363 APT-coupling diversity while preserving the intended causal structure.

1364 The simulator records two synchronized outputs: (1) rendered video frames and (2) frame-level
1365 ground-truth state traces, including object identity, pose, linear velocity, angular velocity, contact
1366 pairs, contact forces, support relations, material parameters, and surface-region identifiers. The
1367 intended coupling rule is only a causal prior. The final labels are assigned from the realized trace, so
1368 failed executions can be filtered, unintended outcomes can be relabeled, and extra realized transitions
1369 can be retained when supported by the trace.

1370 C.7 Step 5: VLM Post-Render Semantic Grounding

1371 Because domain randomization chooses concrete assets during simulation, the abstract template
1372 does not know the final visible objects. We therefore use Qwen3.5-VL after rendering to ground
1373 object slots into concrete visual descriptions. The VLM identifies object category, color, material
1374 appearance, location, and visible relations. It is not asked to infer APT types or timestamps.

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

```

1375 You are given frames from a rendered physics simulation and the abstract scene template below.
1376
1377 Abstract slots:
1378   support: a static support surface
1379   mover: a dynamic rigid object sliding on the support
1380   cover: a garment released above the mover
1381
1382 Task:
1383   Identify the concrete visible object corresponding to each slot.
1384   Return only JSON with:
1385     slot
1386     object_category
1387     color
1388     material_appearance
1389     location
1390     visible_relation
1391
1392 Important:
1393   Do not infer APT labels.
1394   Do not infer timestamps.
1395

```

```

1396 Do not describe invisible simulator states.
1397 Only ground the abstract template into visible objects and relations.

```

Listing 7: Demo VLM answer: semantic grounding.

```

1399 {
1400   "support": {
1401     "object_category": "coffee table",
1402     "color": "brown",
1403     "material_appearance": "wood-like",
1404     "location": "center of the scene",
1405     "visible_relation": "supports the moving object and later the garment"
1406   },
1407   "mover": {
1408     "object_category": "cube",
1409     "color": "blue",
1410     "material_appearance": "matte rigid object",
1411     "location": "on top of the table",
1412     "visible_relation": "slides from left to right across the table"
1413   },
1414   "cover": {
1415     "object_category": "pair of pants",
1416     "color": "red",
1417     "material_appearance": "soft textile garment",
1418     "location": "above the table, then on or near the cube",
1419     "visible_relation": "falls downward and contacts or covers the cube"
1420   }
1421 }
1422 }

```

1424 C.8 Step 6: LLM/Parser GT-Trace-Based APT Stage Parsing

1425 Precise APT timing is obtained from simulator traces, not from visual guessing. We use
 1426 qwen/Qwq-32B as an LLM-assisted structured parser. The parser receives the APT taxonomy,
 1427 the human-defined coupling rule, the VLM-grounded object names, and a compact trace summary. It
 1428 converts numeric state changes into realized APT instances. The intended template is treated as a
 1429 prior, not a label guarantee.

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

```

1430 You are converting simulator ground-truth traces into Atomic Physical Transition annotations.
1431
1432 Important rules:
1433   Use simulator traces as the source of timing and physical state.
1434   Use VLM grounding only for object names and visual descriptions.
1435   The intended template is a prior, not a label guarantee.
1436   If the realized dynamics differ from the intended template, relabel according to the trace.
1437   Output concise evidence, not chain-of-thought.
1438
1439 APT definitions:
1440   contact_init: free then contact; a new object-object or object-surface contact begins.
1441   contact_break: contact then free; an existing contact pair separates.
1442   elastic_rebound: approach then retreat; post-contact relative velocity reverses with rebound.
1443   inelastic_capture: moving then captured/resting; object remains in contact without rebound.
1444   free_fall_onset: supported then falling; object becomes unsupported and moves downward.
1445   projectile_apex: ascending then descending; vertical velocity crosses from positive to negative.
1446   landing: falling then supported; falling object first becomes supported by a surface/object.
1447   static_breakaway: stationary then sliding; object begins sliding after force overcomes static friction.
1448   sliding_arrest: sliding then stationary; sliding object comes to rest and remains below threshold.
1449   surface_transition: moving object crosses from one surface/material region to another.
1450   rotation_onset: non-rotating then rotating; angular velocity becomes visible/significant.
1451   toppling: stable then unstable; object loses stability and begins falling over.
1452   rotation_arrest: rotating then stationary; angular velocity damps below threshold.
1453   settling: object motion damps to stable rest after wobble, bounce, sliding, or deformation.
1454
1455 Grounded slots:
1456   mover = blue cube
1457   support = brown wooden table
1458   cover = red pants
1459
1460 Trace summary:
1461   fps = 60
1462
1463   t=0.00s:
1464     blue_cube supported_by table = true
1465     blue_cube speed = 0.00 m/s

```

```

1467
1468 t=0.15s:
1469     blue_cube horizontal impulse applied
1470     blue_cube speed increases from 0.00 to 1.08 m/s
1471
1472 t=0.44s:
1473     blue_cube x crosses surface_region low_friction -> high_friction
1474     speed begins decreasing more rapidly
1475
1476 t=0.81s:
1477     blue_cube speed < 0.03 m/s for 0.20s
1478     blue_cube remains supported_by table = true
1479
1480 t=0.88s:
1481     red_pants release flag changes fixed -> dynamic
1482     red_pants supported = false
1483     red_pants vertical velocity becomes negative
1484
1485 t=1.18s:
1486     contact_pair(red_pants, blue_cube) changes false -> true
1487     contact_force normal > threshold
1488     red_pants vertical velocity magnitude decreases sharply
1489
1490 t=1.25s:
1491     red_pants remains in contact with blue_cube/table
1492     no rebound; relative normal velocity does not become positive
1493
1494 t=1.70s:
1495     red_pants particle velocities < threshold for 0.30s
1496     blue_cube speed remains < threshold
1497
1498 Output format:
1499     A JSON array.
1500     Each record must contain:
1501         type
1502         t_ms
1503         objects
1504         evidence
1505

```

Listing 9: Demo LLM answer: realized APT chain.

```

1506 [
1507 {
1508     "type": "static_breakaway",
1509     "t_ms": 150,
1510     "objects": ["blue cube"],
1511     "evidence": "blue cube changes from rest to horizontal sliding after impulse"
1512 },
1513 {
1514     "type": "surface_transition",
1515     "t_ms": 440,
1516     "objects": ["blue cube", "brown wooden table"],
1517     "evidence": "blue cube crosses from low-friction to high-friction surface region"
1518 },
1519 {
1520     "type": "sliding_arrest",
1521     "t_ms": 810,
1522     "objects": ["blue cube", "brown wooden table"],
1523     "evidence": "blue cube speed stays below threshold while supported by table"
1524 },
1525 {
1526     "type": "free_fall_onset",
1527     "t_ms": 880,
1528     "objects": ["red pants"],
1529     "evidence": "red pants become unsupported and acquire downward velocity"
1530 },
1531 {
1532     "type": "contact_init",
1533     "t_ms": 1180,
1534     "objects": ["red pants", "blue cube"],
1535     "evidence": "new contact pair between red pants and blue cube appears"
1536 },
1537 {
1538     "type": "landing",
1539     "t_ms": 1180,
1540     "objects": ["red pants", "blue cube"],
1541     "evidence": "falling red pants first become supported through contact"
1542 },
1543 {
1544     "type": "inelastic_capture",
1545

```

```

1546     "t_ms": 1250,
1547     "objects": ["red pants", "blue cube"],
1548     "evidence": "red pants remain in contact without rebound after impact"
1549   },
1550   {
1551     "type": "settling",
1552     "t_ms": 1700,
1553     "objects": ["red pants", "blue cube"],
1554     "evidence": "garment and cube remain below motion thresholds for the hold window"
1555   }
1556 ]

```

1558 This example illustrates why rule-guided simulation produces more compositionally dense APT data
1559 than existing event-centered clips. A conventional falling or bouncing clip may contain only two or
1560 three obvious transitions. In contrast, the coupled friction-contact-garment scene produces a longer
1561 chain involving sliding, surface transition, arrest, falling, contact, landing, capture, and settling.

1562 C.9 Additional Demo: Gravity-Contact Rigid-Body Scene

1563 We also construct simpler rigid-body templates to cover gravity, contact, landing, inelastic capture,
1564 and settling. The example below follows the same rule-guided pipeline but uses only static rigid
1565 supports and dynamic rigid objects.

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

```

1566 Human coupling rule:
1567 Gravity, contact, landing, inelastic capture, and settling can co-occur.
1568 Use only static rigid supports and dynamic rigid objects.
1569 Construct a category-level scene where multiple small rigid objects fall onto a table
1570 and may contact one another after landing.
1571
1572 Output:
1573   object slots
1574   scene skeleton
1575   expected APTs
1576   success checks
1577   likely failure cases
1578

```

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

```

1580 {
1581   "template_name": "multi_rigid_drop_on_table",
1582   "object_slots": {
1583     "support": "static_rigid table",
1584     "falling_object_A": "dynamic_rigid object",
1585     "falling_object_B": "dynamic_rigid object"
1586   },
1587   "scene_skeleton": [
1588     "Two dynamic rigid objects are initialized above a table.",
1589     "They fall under gravity.",
1590     "Each object lands on the table.",
1591     "One object may roll or slide into the other.",
1592     "Both objects settle on the table."
1593   ],
1594   "expected_apts": [
1595     "free_fall_onset",
1596     "landing",
1597     "contact_init",
1598     "inelastic_capture",
1599     "sliding_arrest",
1600     "settling"
1601   ],
1602   "success_checks": [
1603     "each dynamic object has downward velocity before first table contact",
1604     "each object creates a new contact pair with the table",
1605     "at least one post-landing object-object contact occurs or the scene is relabeled",
1606     "all dynamic objects satisfy final rest thresholds"
1607   ],
1608   "likely_failure_cases": [
1609     "one object misses the table",
1610     "objects bounce too strongly and leave the view",
1611     "objects never contact each other",
1612     "final settling is not reached within the clip"
1613   ]
1614 }
1615

```

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

```

1617 scene_name: multi_rigid_drop_on_table
1618 description: >
1619     Two dynamic rigid objects fall onto a table, land, may collide with
1620     each other, and finally settle.
1621
1622 background:
1623     intensity: [1000, 2000]
1624     texture:
1625         - "~/Background/champagne_castle_1_4k.hdr"
1626         - "~/Background/evening_road_01_4k.hdr"
1627         - "~/Background/noon_grass_4k.hdr"
1628         - "~/Background/qwantani_4k.hdr"
1629
1630 light:
1631     light_1:
1632         types: ["Rect", "Sphere", "Cylinder"]
1633         common:
1634             initial_pos_range: [-0.2, -0.2, 0.0, 0.2, 0.5, 0.0]
1635             initial_ori_range: [-30, -30, -30, 30, 30, 30]
1636             color_temperature_range: [3000, 9000]
1637             color_range: [[0.8, 0.8, 0.8], [1.2, 1.2, 1.2]]
1638         Rect:
1639             intensity_range: [2000, 5000]
1640             width_range: [1.0, 3.0]
1641             height_range: [1.0, 3.0]
1642         Sphere:
1643             intensity_range: [5000, 10000]
1644             radius_range: [0.1, 0.5]
1645         Cylinder:
1646             intensity_range: [4000, 8000]
1647             radius_range: [0.1, 0.3]
1648             length_range: [1.0, 2.0]
1649
1650 camera:
1651     random: true
1652     look_at: [0.2, 0.4, 0.75]
1653     pos_range: [-1.5, -2.0, 1.2, 1.5, -1.0, 2.2]
1654     focal_length_range: [24, 45]
1655
1656 objects:
1657     simple_desk:
1658         usd:
1659             - "~/Desk/Collected_appleseed_coffeetable/appleseed_coffeetable_inst_base.usd"
1660             - "~/Desk/Collected_cline/cline_inst_base.usd"
1661             - "~/Desk/Collected_oaktablesmall/oaktablesmall_inst_base.usd"
1662             - "~/Desk/Collected_willowbench/willowbench_inst_base.usd"
1663         random: true
1664         default_material: true
1665         num_per_env: 1
1666         semantic_label: "static_support_surface"
1667         simple_desk_1:
1668             pos: [0.0, 0.0, 0.0]
1669             ori: [0.0, 0.0, 90.0]
1670             visual:
1671                 scale: [1.3, 1.3, 1.0]
1672                 color:
1673                     - [0.8, 0.7, 0.6]
1674                     - [0.6, 0.5, 0.4]
1675                     - [0.4, 0.3, 0.2]
1676                     - [0.2, 0.1, 0.0]
1677                 visual_material:
1678                     material_usd_folder: "~/Material/Garment"
1679             physics:
1680                 type: "geometry"
1681                 collision: true
1682                 physics_material:
1683                     static_friction_range: [0.5, 1.0]
1684                     dynamic_friction_range: [0.4, 0.9]
1685                     restitution_range: [0.0, 0.15]
1686
1687     falling_rigid_A:
1688         common:
1689             primitive_scale: [0.30, 0.30, 0.30]
1690             initial_pos_range: [-0.35, -0.20, 1.30, -0.05, 0.20, 1.80]
1691             initial_ori_range: [-30, -30, -30, 30, 30, 30]
1692         usd:
1693             - "~/Object/Mug"
1694             - "~/Object/apple"
1695             - "~/Object/Sugar_Box"

```

```

1697     random: true
1698     num_per_env: 1
1699     semantic_label: "falling_rigid_A"
1700     falling_rigid_A_1:
1701         visual:
1702             scale: [1.0, 1.0, 1.0]
1703             visible: true
1704             color:
1705                 - [1.0, 0.0, 0.0]
1706                 - [0.0, 0.0, 1.0]
1707                 - [1.0, 1.0, 0.0]
1708                 - [0.0, 1.0, 1.0]
1709             visual_material:
1710                 material_usd_folder: "~/Material/Garment"
1711     physics:
1712         type: "dynamic"
1713         collision: true
1714         mass_range: [0.2, 0.7]
1715         ratio: 1.1
1716         physics_material:
1717             static_friction_range: [0.2, 0.6]
1718             dynamic_friction_range: [0.2, 0.6]
1719             restitution_range: [0.0, 0.20]
1720             linear_velocity_range:
1721                 - [0.05, 0.00, 0.00]
1722                 - [0.25, 0.10, 0.00]
1723             angular_velocity_range:
1724                 - [-0.2, -0.2, -0.2]
1725                 - [0.2, 0.2, 0.2]
1726             track_contact_forces: true
1727             prepare_contact_sensor: true
1728
1729     falling_rigid_B:
1730         common:
1731             primitive_scale: [0.30, 0.30, 0.30]
1732             initial_pos_range: [0.05, 0.00, 1.25, 0.35, 0.30, 1.75]
1733             initial_ori_range: [-30, -30, -30, 30, 30, 30]
1734         usd:
1735             - "~/Object/Mug"
1736             - "~/Object/apple"
1737             - "~/Object/Sugar_Box"
1738         random: true
1739         num_per_env: 1
1740         semantic_label: "falling_rigid_B"
1741         falling_rigid_B_1:
1742             visual:
1743                 scale: [1.0, 1.0, 1.0]
1744                 visible: true
1745                 color:
1746                     - [0.0, 1.0, 0.0]
1747                     - [1.0, 0.0, 1.0]
1748                     - [0.0, 1.0, 1.0]
1749                     - [1.0, 0.5, 0.0]
1750             visual_material:
1751                 material_usd_folder: "~/Material/Garment"
1752         physics:
1753             type: "dynamic"
1754             collision: true
1755             mass_range: [0.2, 0.7]
1756             ratio: 1.1
1757             physics_material:
1758                 static_friction_range: [0.2, 0.6]
1759                 dynamic_friction_range: [0.2, 0.6]
1760                 restitution_range: [0.0, 0.20]
1761             linear_velocity_range:
1762                 - [-0.25, -0.10, 0.00]
1763                 - [-0.05, 0.00, 0.00]
1764             angular_velocity_range:
1765                 - [-0.2, -0.2, -0.2]
1766                 - [0.2, 0.2, 0.2]
1767             track_contact_forces: true
1768             prepare_contact_sensor: true
1769
1770     trace_logging:
1771         fps: 60
1772         fields:
1773             - pose
1774             - linear_velocity
1775             - angular_velocity
1776             - contact_pairs
1777             - contact_forces

```

```

1778   - support_relation
1779   - object_identity
1780   - material_parameters
1781
1782 success_checks:
1783   intended_domains:
1784     - gravity
1785     - contact
1786     - friction
1787     - settling
1788 required_conditions:
1789   object_A_falls:
1790     object: "falling_rigid_A"
1791     min_downward_velocity: 0.2
1792   object_B_falls:
1793     object: "falling_rigid_B"
1794     min_downward_velocity: 0.2
1795   table_landing_A:
1796     contact_pair: ["falling_rigid_A", "simple_desk"]
1797   table_landing_B:
1798     contact_pair: ["falling_rigid_B", "simple_desk"]
1799   final_settling:
1800     objects: ["falling_rigid_A", "falling_rigid_B"]
1801     linear_speed_below: 0.03
1802     angular_speed_below: 0.10
1803     hold_duration_sec: 0.30
1804   optional_conditions:
1805     secondary_contact:
1806       contact_pair_any:
1807         - ["falling_rigid_A", "falling_rigid_B"]
1808   failure_filters:
1809     - "one rigid object misses the support surface"
1810     - "object bounces out of camera view"
1811     - "final settling not reached within clip"
1812     - "severe interpenetration"
1813

```

1814 C.10 Step 7: Human Validation and Prompt/Template Refinement

1815 The automatic construction pipeline is validated against the human-labeled anchor. For a subset
1816 of simulated clips, humans annotate the visible APT chain directly from the rendered video. We
1817 compare human annotations with automatically produced chains using type equality and timestamp
1818 tolerance. Disagreements are used to refine four components: (1) coupling rules, when a proposed
1819 scene is physically unnatural; (2) YAML success checks, when the intended transition does not
1820 reliably occur; (3) VLM grounding prompts, when object identity or relation is ambiguous; and (4)
1821 GT-trace parsing thresholds, when simulator state changes occur earlier or later than human-visible
1822 transitions. After calibration, the automatic pipeline produces APT chains that closely match human
1823 annotations while scaling beyond manual labeling.

Listing 13: Prompt for validation disagreement analysis.

```

1824 You are given:
1825   (1) a human-labeled APT chain,
1826   (2) an automatically produced APT chain,
1827   (3) the simulator trace summary,
1828   (4) the rendered-video grounding.
1829
1830 Compare the human and automatic chains.
1831 For each mismatch, classify the likely cause as one of:
1832   template_issue
1833   success_check_issue
1834   grounding_issue
1835   trace_threshold_issue
1836   human_ambiguity
1837   extra_realized_transition
1838
1839 Suggest one concrete fix.
1840 Do not change labels unless the simulator trace supports the change.
1841

```

Listing 14: Demo validation answer.

```

1843 [
1844   {
1845     "mismatch": "automatic contact_init at 1140 ms; human contact_init at 1180 ms",
1846     "cause": "trace_threshold_issue",
1847

```

```

1848   "analysis": "contact force is nonzero before contact is visually clear",
1849   "fix": "increase contact-force threshold or require 2 consecutive frames above threshold"
1850 },
1851 {
1852   "mismatch": "automatic omitted landing for red pants",
1853   "cause": "success_check_issue",
1854   "analysis": "parser produced contact_init but did not add landing for an object that was falling
1855 immediately before first support contact",
1856   "fix": "if an object has negative vertical velocity before first support contact, add landing at the
1857 first support contact"
1858 },
1859 {
1860   "mismatch": "automatic detected extra surface_transition",
1861   "cause": "extra_realized_transition",
1862   "analysis": "simulator trace shows mover crossed from low-friction to high-friction region and the
1863 speed decay changed visibly",
1864   "fix": "keep the label when the motion change is visible; otherwise store it as metadata-only"
1865 }
1866 ]

```

1868 C.11 Step 8: Multi-Format APT Training Data

1869 Each accepted clip is exported into multiple aligned formats. This is important because single-format
1870 APT JSON training can make the model an APT-format specialist and degrade ordinary event-level
1871 video answering. The same physical scene is therefore represented as: (1) APT JSON, (2) grounded
1872 object description, (3) causal transition description, (4) event-level QA, and (5) multiple-choice QA.
1873 These aligned formats allow APT-Tune to learn APTs as reusable causal physical representations
1874 rather than a single output style.

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

```

1875 APT JSON:
1876 [
1877   {
1878     "type": "static_breakaway", "t_ms": 150, "objects": ["blue cube"]},
1879     {"type": "surface_transition", "t_ms": 440, "objects": ["blue cube"]},
1880     {"type": "sliding_arrest", "t_ms": 810, "objects": ["blue cube"]},
1881     {"type": "free_fall_onset", "t_ms": 880, "objects": ["red pants"]},
1882     {"type": "contact_init", "t_ms": 1180, "objects": ["red pants", "blue cube"]},
1883     {"type": "landing", "t_ms": 1180, "objects": ["red pants", "blue cube"]},
1884     {"type": "inelastic_capture", "t_ms": 1250, "objects": ["red pants", "blue cube"]},
1885     {"type": "settling", "t_ms": 1700, "objects": ["red pants", "blue cube"]}
1886   ]
1887
1888 Grounded object description:
1889   A blue cube slides across a brown wooden table, slows down on a rougher region,
1890   and stops. A red pair of pants falls from above, lands on the cube, remains in
1891   contact, and settles.
1892
1893 Causal transition description:
1894   The rigid object first breaks away from rest and slides. Its motion changes when
1895   it enters a higher-friction surface region, and friction arrests the sliding.
1896   The garment then becomes unsupported, falls under gravity, initiates contact with
1897   the cube, is captured without rebound, and settles.
1898
1899 Event-level QA:
1900   Q: What happens in the video?
1901   A: A blue cube slides on a table and stops, then a red garment falls onto it and settles.
1902
1903 Multiple-choice QA:
1904   Q: Which description best matches the video?
1905   A. The cube bounces elastically off a wall.
1906   B. The cube slides to a stop, and a garment falls onto it and settles.
1907   C. The garment flies upward after hitting the cube.
1908   D. The table topples over.
1909
1910 Correct answer: B.

```

1912 C.12 Dataset Split and Evaluation Protocol

1913 For the human-labeled data, the train/test split is stratified by source and scenario family using seed
1914 7. The training split contains 997 trials, approximately 21.8K APT instances, consisting of 400
1915 CLEVRER trials and 597 Physion++ trials. The test split contains 249 trials, approximately 5.5K
1916 APT instances, consisting of 100 CLEVRER trials and 149 Physion++ trials. 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 therefore report recall and provide per-type breakdowns.

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 16 uniformly sampled frames per video at source resolution. Models with separate frame-rate conventions receive matched 16-frame samples.
2. **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.
3. **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.
4. **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.
5. **Controlled comparison.** The same prompt and frame list are used for zero-shot baselines and fine-tuned models. The only difference across rows in the result table is the model weights.

Listing 16: Evaluation-time APT-schema prompt.

```

You are given 16 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}
]
```