WHERE DID THE AGENT GO WRONG? LOCALIZING—AND FIXING—FAILURE STEPS IN MULTI-STEP VIDEO-EDITING AGENT TRACES

Anonymous authors
Paper under double-blind review

ABSTRACT

Autonomous agents run long, tool-rich workflows, yet when a run fails, telling *where* it went wrong is hard: a syntactically valid trace can quietly drift off the user's goal. We study this *failure-step localization* problem in multi-step video editing, where one request expands into a chain of metadata probes, shot detections, clip cuts, and concatenations. We introduce AV-FAIL, a benchmark that asks a judge to identify the *earliest* faulty step of an agent trace. AV-FAIL couples a *controlled* track injecting one of five fault types (or none) for programmatic exact-step gold labels with a *real* track of $1{,}848$ genuine runs verified via multi-model human-in-the-loop review. A blind two-annotator study fixes the *human ceiling* at Cohen's $\kappa = 0.63$ (substantial), which *no* automatic judge reaches: the best single model attains only $\kappa = 0.48$ and different judges err in *opposite* directions, so automatic failure localization cannot stand in for human review. Across five judge models we find a sharp *difficulty spectrum*: mechanical faults are nearly solved (1.00), a *cumulative-budget* fault is a systematic blind spot (0.20), and clean traces trigger *over-attribution* (a 0.35 false-positive rate); both effects are strongly model-dependent. Two lightweight mitigations each help but interfere when combined. We resolve this with *Trace-State Prompting*, which—wherever a constraint is expressible as a checkable predicate over the trace—computes that trace state deterministically and routes only the residual decision to the model, raising accuracy from 0.775 to 0.872 and closing the budget blind spot losslessly. On real traces it surfaces 57 semantic failures that objective signals miss, showing *off-goal* drift—not crashes—is the dominant unhandled failure mode. The findings are not video-specific: ported to a data-pipeline (ETL) agent domain, the same ordering recurs and the protocol again lifts accuracy from 0.658 to 0.883. AV-FAIL provides a reproducible lens on a capability that agentic evaluation has so far left largely unmeasured.

1 INTRODUCTION

Large language model (LLM) agents now routinely orchestrate dozens of tool calls for a single instruction (Yao et al., 2023; Schick et al., 2023; Qin et al., 2024), yet evaluation remains largely *outcome-oriented* (Jimenez et al., 2024; Zhou et al., 2024; Mialon et al., 2024): did the final artifact satisfy the request? This coarse view hides the question that arises the moment a run fails—*where did the agent go wrong?* Localizing the earliest faulty step separates an actionable failure report from an opaque one and is the prerequisite for automated repair, targeted scaffolding, credit assignment in reinforcement learning (Lightman et al., 2024), and trustworthy oversight.

Localization is under-measured because it is hard in a way outcome metrics cannot expose. A capable agent can produce a trace that is *locally* plausible at every step—each call type-checks and returns without error—while the trajectory drifts from the user's intent, leaving no crash and no anchor for an automated checker. Conversely, on a *correct* run a judge may hallucinate a culprit, over-attributing failure to a benign step. Both pathologies are invisible to end-to-end success rates.

We instantiate this in **multi-step video editing**, unusually well suited to stress-test localization: a single request ("make a 20s highlight of the three chase scenes, in order") expands into heteroge-

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

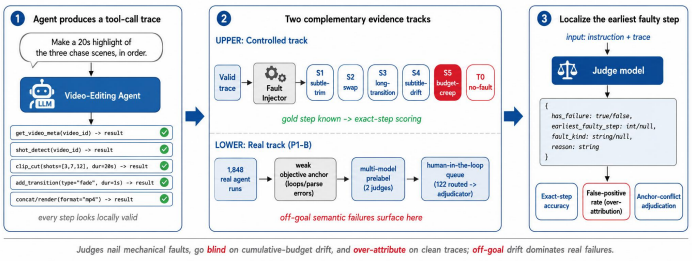


Figure 1: Overview of AV-FAIL: a video-editing agent produces a tool-call trace for a natural-language editing request; we inject one of five controllable fault types (or none) into an otherwise-valid trace, and ask a judge model to localize the *earliest* faulty step. Programmatic gold labels enable exact-step scoring. A parallel real-trace track (P1-B) applies the same localization task to a pool of 1,848 genuine agent runs; two judges prelabel 1,617 of them, and a 268-run adjudication pilot routes the ambiguous cases to a high-capability adjudicator with human spot-checks.

neous operations—metadata probing, shot detection, clip cutting, subtitle handling, concatenation—whose correctness depends on *ordering*, *timing*, and a *global budget* (total duration), not merely on each call succeeding, giving a rich space of semantically distinct ways to fail.

Our benchmark, AV-FAIL, poses one task: given an agent trace, name the *earliest* faulty step (or declare it fault-free). The **controlled** track injects exactly one of five fault types—subtle over-trim (S1), adjacent-clip swap (S2), over-long transition (S3), subtitle drift (S4), cumulative-budget creep (S5)—or no fault (T0) into valid traces, so gold labels are known by construction and scoring is exact. The **real** track (P1-B) applies the identical task to 1,848 genuine runs, using weak objective anchors plus a multi-model, human-in-the-loop protocol to establish reference labels where no programmatic gold exists.

Evaluating five judge models yields three findings. **(F1) A steep difficulty spectrum.** Mechanical faults are essentially solved (S2$=1.00$, S3$=1.00$, S4$=0.96$), but cumulative-budget S5 collapses to 0.20: judges fail to integrate per-clip durations into a running total and overwhelmingly declare "no fault"—a genuine blind spot, not random error. **(F2) Over-attribution.** On fault-free traces (T0) models invent a culprit 35% of the time, a strongly model-dependent false-positive rate. **(F3) Targeted mitigations trade off.** A budget scaffold raises S5 by $+0.51$ and a no-fault gate raises T0 by $+0.17$, but each has side effects on neighboring faults (the gate depresses S4). On the real track our protocol surfaces 57 semantic failures that objective signals miss, confirming off-goal drift—not crashes—is the dominant unhandled failure mode.

Contributions. (i) AV-FAIL, the first benchmark to isolate *failure-step localization* for multi-step tool-using agents, pairing a controlled track (exact gold-step labels) with a $1{,}848$-run real track. (ii) A characterization of *why* localization is hard, distinguishing a cumulative-budget blind spot (F1) from over-attribution (F2), both quantified and model-dependent. (iii) Two targeted mitigations quantified for gains and side effects, showing the gap is addressable but interference arises when they are naively combined. (iv) *Trace-State Prompting*, which computes checkable trace state deterministically and routes only the residual to the model, dominating every single-prompt alternative: it raises overall localization $0.775 \rightarrow 0.872$ (paired $p < 10^{-7}$) and closes the budget blind spot losslessly ($0.31 \rightarrow 0.89$). We release the data, fault injectors, judge harness, and protocol to make localization a first-class, reproducible axis of agentic evaluation.

2 RELATED WORK

Agentic benchmarks. A growing body of benchmarks evaluates tool-using LLM agents on task *completion*—web navigation and realistic web trajectories (Zhou et al., 2024; Yao et al., 2022; Deng et al., 2023), software engineering (Jimenez et al., 2024; Yang et al., 2024), computer control (Xie et al., 2024), API orchestration and stable tool-use evaluation (Qin et al., 2024; Li et al., 2023; Guo et al., 2024; Huang et al., 2024b), and general and user-interactive agent tasks (Mialon et al., 2024; Liu et al., 2024; Ma et al., 2024; Yao et al., 2024); see Wang et al. (2024a) for a survey. Built on tool-use paradigms (Yao et al., 2023; Schick et al., 2023), these score an end-to-end success rate.

Type	Fault	Reasoning demanded
S1	subtle over-trim	local (magnitude)
S2	adjacent swap	local (order)
S3	over-long transition	local (single-step arg)
S4	subtitle drift	local (timing)
S5	**cumulative-budget creep**	**global integration**
T0	**no fault**	**restraint (answer "none")**

Table 1: The six controlled conditions. S1–S4 are step-locally visible; S5 requires summing clip durations over the whole trace against the global budget, and T0 has no injected error (correct answer: "no failure").

AV-FAIL is orthogonal: it localizes the earliest faulty step, shifting evaluation from outcome to trajectory structure.

Process evaluation and self-correction. Process-supervision and step-level reward models score intermediate reasoning (Lightman et al., 2024; Uesato et al., 2022; Wang et al., 2024c), and "LLM-as-judge" methods assess trajectories holistically (Zheng et al., 2023; Li et al., 2024; Gu et al., 2024) despite systematic biases (Wang et al., 2024b); work on self-refinement (Madaan et al., 2023), reflection (Shinn et al., 2023), and error taxonomy (Cemri et al., 2025) studies recovery once a failure is noticed, but Huang et al. (2024a) show self-correction is unreliable. We differ in requiring *exact-step* attribution against gold labels and in constructing faults whose localization demands *global* reasoning (Section 5), repurposing video-editing agents whose correctness hinges on ordering, timing, and global budgets (extended discussion in Appendix I).

Failure attribution benchmarks. Closest are two efforts that attribute failures inside agent runs. Cemri et al. (2025) (MAST) build a taxonomy of *why* multi-agent systems fail but do not require outputting a single decisive step; Zhang et al. (2025) (Who&When) asks which agent failed and when in 184 multi-agent logs. AV-FAIL differs on three decisive axes: (i) *controlled negatives*—we inject each fault at a known step and add a no-fault track (T0), giving an objective gold step and a direct false-positive measurement, unlike prior sets that label naturally occurring failures; (ii) *reasoning-typed difficulty* that isolates local pattern-matching from global integration, revealing the cumulative-budget blind spot (S5) aggregate accuracy hides; and (iii) *a paired real track* of $1{,}848$ runs quantifying transfer.

3 TASK, BENCHMARK, AND PROTOCOL

Setup. An agent executes a natural-language editing instruction over a fixed toolset (metadata probing, shot detection, loudness analysis, clip cutting, subtitle operations, concatenation), yielding a *trace* $\tau = (s_0, \ldots, s_{n-1})$; each step s_i records the tool, arguments, status, truncated result, and elapsed time. A *judge* model must decide whether τ contains a failure and, if so, name the *earliest* faulty step.

Fault types. The controlled track defines five injected faults plus a no-fault control (Table 1), spanning mechanical-to-semantic difficulty. The extremes are a purely *global* fault (S5) and a *negative* control (T0), probing integration and restraint—capabilities step-local judging cannot supply.

Scoring. We score *exact-step* accuracy: a prediction is correct iff it names the injected step ($\text{pred} = -1$ encodes "no failure"). For S5 we add a *double-layer* breakdown—strict localization ($\text{pred} = \text{gold}$), *blind* miss ($\text{pred} = -1$), and mislocalization; for T0 we report the false-positive rate $\Pr[\text{pred} \neq -1]$. We report Wilson 95% intervals for proportions and bootstrap intervals ($B = 2000$) for aggregate deltas.

3.1 CONSTRUCTION AND EVALUATION PROTOCOL

Controlled track. A deterministic injector applies exactly one of S1–S5 to a valid trace (recording the injected step as gold); T0 traces are untouched with gold $= -1$. Gold labels are thus exact and

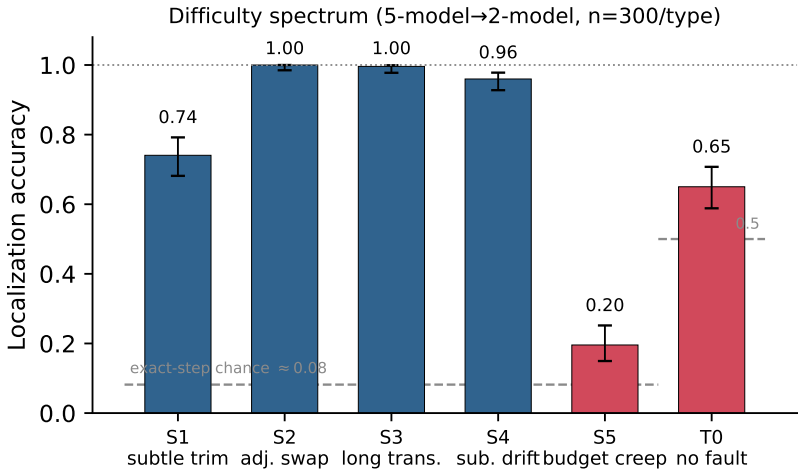


Figure 2: **Difficulty spectrum.** Exact-step accuracy by fault type (Wilson 95% intervals). Mechanical faults (S2–S4) are near-solved; the two highlighted hard regimes—budget creep S5 (0.20) and no-fault T0 (0.65)—are analyzed below.

need no human annotation. Each of the six conditions is evaluated over 300 tasks under matched content.

Judge harness. We evaluate five frontier judge models spanning four providers: CLAUDE-OPUS-4-8 (Anthropic Claude Opus 4.8), GEMINI-3.5-FLASH (Google Gemini 3.5 Flash), DEEPSEEK-V4-PRO (DeepSeek V4-Pro), QWEN3.7-MAX (Alibaba Qwen3.7-Max), and DOUBAO-SEED-2.0-PRO (ByteDance Doubao-Seed-2.0-Pro, build 260215). Each receives the instruction and full trace and returns a structured JSON verdict (has-failure, earliest-faulty-step, fault-kind, rationale); malformed outputs are excluded (1460 valid records per condition set). All judges see identical prompts and are queried with greedy decoding (temperature 0; a few reasoning-only endpoints that ignore the temperature field are left at their default). For brevity, later text abbreviates these judges by provider family (CLAUDE, GEMINI, DEEPSEEK, QWEN, DOUBAO); the exact endpoints are the version strings above. All prompts, parsing rules, model identifiers, and the scorer are released.

Mitigations. We add two lightweight, *targeted* prompt interventions: a *budget scaffold* (SCAFFOLD-S5) that maintains a running duration total, and a *no-fault gate* (GATE-T0) adding an explicit "if no step violates the instruction, answer no-failure" guard. Each is *designed* on a 100-task split and evaluated on the full 300-task suite (held-out 200 reported separately in Section 4.1); we report per-fault-type Δacc to expose both gains and side effects.

Real track (P1-B). To test transfer, we assemble 1,848 genuine agent runs and pose the identical task. Lacking programmatic gold, we combine three signals: *(1)* weak objective anchors that flag mechanical anomalies (loops, parse errors) but cannot see semantic *off-goal* drift; *(2)* two judge models prelabeling each run independently; *(3)* human-in-the-loop verification—runs where models disagree, conflict with the anchor, or lack predictions go to a review queue, where a high-capability adjudicator (given the full trace, both verdicts, and the anchor) produces a first-pass reference verdict, spot-checked by a human. Of 268 prelabeled runs, 122 (46%) were routed—exactly where automated signals are insufficient.

4 Experiments and Analysis

We report the controlled track over five judge models. All proportions carry Wilson 95% intervals.

A steep difficulty spectrum. Figure 2 shows a wide spread: adjacent-swap (S2) and over-long transition (S3) are essentially perfect at 1.00, subtitle drift (S4) 0.96, and subtle over-trim (S1) 0.74, whereas the cumulative-budget fault **S5 falls to** 0.20 and the no-fault control **T0 to** 0.65. The 70-

Fault type	n	Accuracy	Wilson 95% CI
S1 subtle over-trim	239	0.741	$[0.68, 0.79]$
S2 adjacent swap	250	1.000	$[0.99, 1.00]$
S3 over-long transition	249	0.996	$[0.98, 1.00]$
S4 subtitle drift	249	0.960	$[0.93, 0.98]$
S5 budget creep	230	**0.196**	$[0.15, 0.25]$
T0 no fault	243	**0.650**	$[0.59, 0.71]$

Table 2: Controlled-track exact-step accuracy (five-model config, 1460 valid verdicts; n is the per-type count after excluding malformed outputs). S5 and T0 are the two hard regimes.

point gap between the easiest and hardest fault shows difficulty is dominated not by the domain but by *what kind* of reasoning a fault demands—local pattern-matching (easy) versus global integration or restraint (hard). Against a chance baseline of ≈ 0.08 (uniform guessing over ~ 12 steps; dashed line), every fault type—including S5—sits well above random, so the difficulty is one of *reliable* localization, not of beating chance.

Statistical protocol. We report Wilson intervals for per-condition proportions and bootstrap intervals ($B = 2000$) for aggregates such as mitigation deltas; malformed judge outputs are excluded rather than scored, keeping denominators honest. Because every condition is evaluated on the *same* tasks, we test paired improvements with McNemar's test (continuity-corrected): Trace-State Prompting over the baseline is significant overall (82 fixed vs. 24 broken, $p = 3.1 \times 10^{-8}$), and closes the S5 blind spot with 58 traces fixed and zero broken ($p = 7.2 \times 10^{-14}$).

4.1 Why Localization Fails

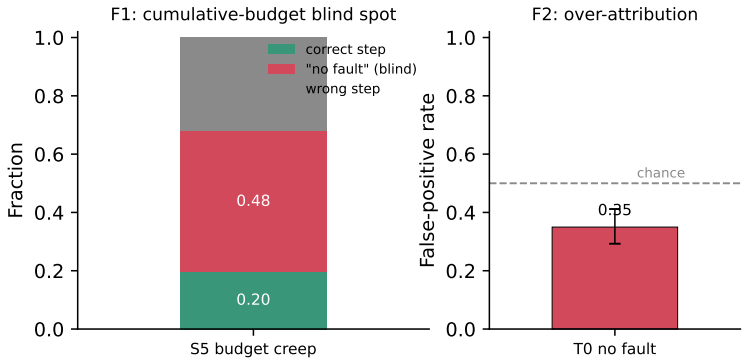
Finding 1: a cumulative-budget blind spot. Figure 3a (left) decomposes S5. The dominant failure mode is not mislocalization but *blindness*: the modal response is "no failure at all." Because no single step violates the instruction, judges that reason step-locally never accumulate the per-clip durations into a running total and so never see the overrun—a qualitative capability gap (integration over the trace) rather than noisy guessing, and exactly the regime our budget scaffold targets.

Finding 2: over-attribution on clean traces. On fault-free traces (Figure 3a, right) judges declare a failure 35% of the time, and the per-model T0 false-positive rate spans a wide range—restraint on clean inputs is *model-dependent*. A benchmark measuring only failure *detection* would miss this; only a negative control exposes it.

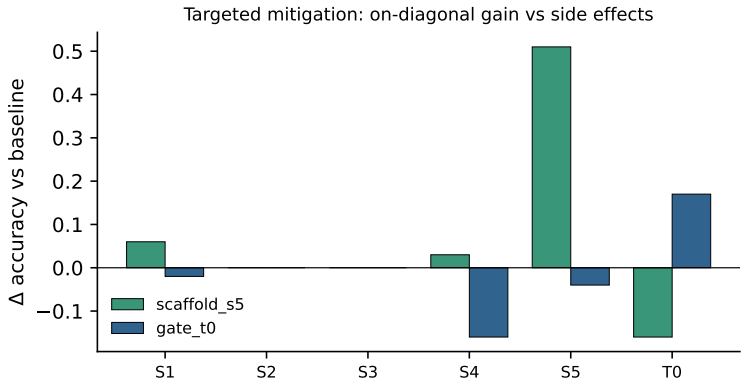
Finding 3: each gap is addressable in isolation, but the fixes conflict. Figure 3b reports each intervention's per-fault delta. The budget scaffold raises S5 by $\Delta = +0.51$ ($0.31 \rightarrow 0.82$ on the two instruction-following judges Claude and Gemini; the 0.31 baseline is this two-judge subset, versus the five-model 0.20 of Table 2, as the mitigations are run only on these two judges) but adds clean-trace false alarms ($\Delta_{\text{T0}} = -0.16$); the no-fault gate raises T0 by $\Delta = +0.17$ yet depresses subtitle-drift localization ($\Delta_{\text{S4}} = -0.16$), as a stronger "no failure" prior suppresses genuine detections. The gains are not overfit (scaffold S5 $+0.53$ design vs. $+0.50$ on 200 held-out tasks; gate T0 $+0.19$ vs. $+0.16$, within bootstrap CI). Both are thus addressable by lightweight prompting, but the fixes pull against each other: naively combining them *lowers* overall accuracy because the gate cancels the scaffold (Section 5), so closing both gaps at once requires separating them by construction—which motivates our method.

Robustness to the trace generator. To rule out that the S5 blind spot is an artifact of our single trace-synthesis agent, we generate 90 tasks with three *different* planners (Claude, DeepSeek, Qwen; 30 each, balanced over the six faults) and re-evaluate with two judges. The spectrum holds regardless of author: S5 is $0.50/0.25/0.15$ across sources (overall 0.23, matching the 0.31 two-judge baseline), mechanical faults stay near-ceiling, and over-attribution persists. The findings reflect a property of the judge models, not of any generator (per-source breakdown in Appendix E).

Finding 4: general reasoning does not close the gaps. A natural question is whether the two hard regimes simply demand *more* reasoning rather than a different mechanism. Appendix F (Table 8)



(a) **The two hard regimes.** *Left (S5):* judges rarely localize the true step (green), most giving a *blind* "no failure" verdict (red) or mislocalizing (gray). *Right (T0):* on clean traces judges invent a culprit at a 0.35 false-positive rate (coin-flip 0.5 dashed).



(b) **Targeted mitigations trade off.** Per-fault accuracy change vs. baseline (300-task set, claude/gemini). The scaffold lifts S5 ($+0.51$) but raises clean-trace false alarms ($\Delta_{T0} = -0.16$); the gate raises T0 ($+0.17$) but depresses S4 ($\Delta_{S4} = -0.16$). Absolutes in Table 3.

tests three off-the-shelf test-time-reasoning methods—self-consistency (B4) (Wang et al., 2023), self-refine (B5) (Madaan et al., 2023), and stepwise PRM-style scoring (B6) (Lightman et al., 2024; Wang et al., 2024c)—against the same two judges as B1. None closes the S5 integration blind spot: majority voting leaves it at 0.26 (not significant overall, $p = 0.20$), while self-refine and stepwise scoring *drop below* the 0.31 baseline (0.17 and 0.24), and stepwise scoring even lowers overall accuracy significantly ($p = 1.4 \times 10^{-7}$, paired McNemar). Aggregating or iterating step-local verdicts cannot manufacture the global running total that S5 requires—confirming the gap is a mechanism deficit, not a compute deficit, and motivating the deterministic routing of Trace-State Prompting.

5 Trace-State Prompting

Section 4.1 isolates two orthogonal weaknesses: an *integration* failure on S5, where a step-local judge never forms the running total needed to see the overrun, and an *over-attribution* failure on T0. Naively combining the budget scaffold and no-fault gate in one prompt *lowers* accuracy below the scaffold alone—the gate's "prefer no-failure" prior cancels the scaffold's budget-flagging. **Trace-State Prompting** (TSP) instead separates the concerns by *routing* rather than *stacking*: it computes the model-independent, machine-checkable parts of the trace state before the model is consulted.

Stage 1: deterministic trace-state check. The cumulative-budget constraint is arithmetic: the timeline length $\sum_i (\text{out}_i - \text{in}_i)$ over `trim` steps plus $\sum_j \text{dur}_j$ over `transition` steps is faulty exactly when it exceeds `spec.max_total_ms`. TSP computes this total directly; when the timeline

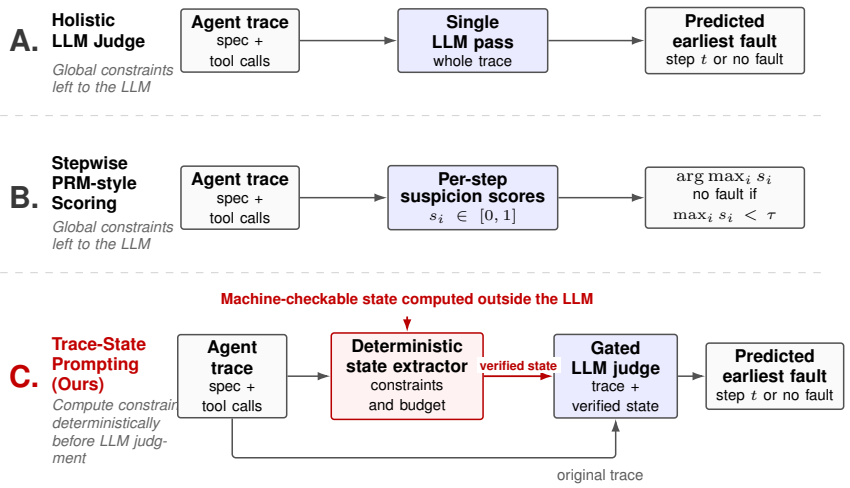


Figure 4: Localization paradigms. Published baselines either judge the whole trace in a single pass (A, holistic LLM-judge (Zheng et al., 2023)) or emit a per-step suspicion score and take its arg-max (B, stepwise PRM-style scoring (Lightman et al., 2024; Wang et al., 2024c)); both leave global, machine-checkable constraints (e.g. cumulative budget) to the model. Our *Trace-State Prompting* (C) instead routes: a deterministic state/budget tool computes the checkable trace state and *gates* a downstream LLM judge, decoupling arithmetic integration from open-ended attribution.

Fault	Base	Scaf.	Gate	Combo	**TSP**
S1 subtle-trim	0.69	0.75	0.67	0.67	0.68
S2 adjacent-swap	1.00	1.00	1.00	1.00	1.00
S3 long-transition	1.00	1.00	1.00	1.00	1.00
S4 subtitle-drift	0.91	0.94	0.75	0.74	0.75
S5 budget-creep	0.31	0.82	0.27	0.66	**0.89**
T0 no-fault	0.74	0.58	0.91	0.81	**0.91**
Overall	0.775	0.848	0.767	0.813	**0.872**

Table 3: **Trace-State Prompting dominates every alternative.** Per-fault accuracy on the 300-task control set, pooled over Claude and Gemini. *Scaf.*: budget scaffold; *Gate*: no-fault gate; *Combo*: naive concatenation. Only TSP lifts both hard regimes *together* (S5 to 0.89; T0 to 0.91) for the best overall (0.872), beating baseline on both judges (Claude $0.850 \rightarrow 0.893$, Gemini $0.697 \rightarrow 0.850$).

is over budget *and* the trims are approximately balanced—the signature of budget *creep* (every step reasonable but jointly overflowing, unlike a single oversized cut)—it emits the last `trim` step as the fault, with no model call. The balance test (trim spread within 15% of the mean) keeps it from firing on the single-oversized-clip fault S1. On the control set this stage fires on 40 of 300 traces and is *correct on all* 40 (39 S5, one correct S1), recovering the integration-hard regime losslessly.

We verify the checks are robust rather than tuned—the 15% tolerance sits on a stable $5\% \rightarrow 15\%$ plateau at precision 1.00, and re-injecting each fault with parameters *disjoint from* the check predicate keeps the order (S2) and timing (S3) checks at $P = R = 1.00$ on all 300 held-out traces (Appendix J). The budget check is where held-out injection bites: naively it drops to precision 0.739 (recall 0.962) because 96 over-transition (S3) traces also overflow the timeline—exactly the overlapping-fault ambiguity that motivates *routing rather than stacking*, since consulting the timing check first (Stage-1 precedence) restores budget-check precision to 1.000 at recall 0.962 (Appendix J, Figure 7), leaving 10 residual misses (3.8%).

Stage 2: gated localization for the rest. Every unresolved trace goes to the judge under the no-fault gate—the exact gate of Section 4.1 (same prompt, model, temperature $= 0$), so TSP uses no information beyond that single-probe condition. Because Stage 1 has already removed the budget-creep case, the gate no longer coexists with a budget-tracking instruction and the interference that sinks the naive combination disappears: the "prefer no-failure" prior now suppresses only genuine over-attribution.

Results. TSP attains the highest overall accuracy in Table 3, 0.872 versus 0.775 for the baseline (paired McNemar $\Delta = +0.097$, 82 corrected/24 broken, $p < 10^{-7}$), improving both hard regimes *simultaneously*: S5 $0.31 \rightarrow 0.89$ ($+0.58$, 58 corrected and *zero* broken, $p < 10^{-13}$; above the scaffold's own 0.82, since the deterministic checker is exact where the prompted judge is merely nudged) and T0 $0.74 \rightarrow 0.91$ ($+0.17$, zero broken). Unlike the naive combination, which drags S5 to 0.66 and still inherits the gate's cost, TSP breaks *no* S5 or T0 case—routing the arithmetic sub-problem to a check removes the prompt interference entirely.

The residual trade-off. TSP is not free of side effects: its second stage carries the no-fault gate, so subtitle-drift S4 falls $0.91 \rightarrow 0.75$ (-0.16; 21 broken vs. 5 fixed) as the stronger prior suppresses subtle detections—the restraint-versus-sensitivity tension relocated, not abolished. The lesson is architectural: parts of a judgment reducible to checkable trace state should be computed, not prompted. The recipe extends beyond budgets—an ordering fault (S2) reduces to "CONCAT order equals narrative order" and a timing fault (S3) to "every TRANSITION lies within its bound." Instantiated as two more checks and applied *untuned* to the 300-task suite, each localizes its target at **precision/recall** 1.00 **with zero misfires** $(50/50)$—a reusable pattern, not a budget-specific trick.

6 Generalization: Cross-Domain and Real Traces

6.1 Cross-Domain Replication

A single-domain benchmark risks conflating a genuine reasoning gap with a task-family quirk. We therefore port the *entire* apparatus—taxonomy, generator, judge, trace-state check—to a structurally different **data-pipeline (ETL) agent** (tools LOAD, FILTER, TRANSFORM, AGGREGATE, WRITE) with an ordering constraint, a per-operator memory cap (≤ 4096 MB), and a cumulative cluster budget (≤ 12000 MB). The six axes map *isomorphically* (S1 partition-skew, S2 stage reorder, S3 over-cap operator, S4 schema drift, S5 memory creep, T0 no-fault). We generate a matched 300-task suite (50 per axis), judged by the same roster (claude, gemini) under an identical prompt.

The difficulty ordering replicates. The cross-domain per-axis baseline accuracy (Appendix G, Table 9) matches the video-editing signature: mechanical faults are easy (S1 0.80, S2 0.75, S3 1.00, S4 0.90) while the two reasoning-heavy conditions collapse—budget creep to $\mathbf{0.30}$ and no-fault to $\mathbf{0.20}$ (a spurious fault on $80/100$ clean traces)—with the same per-model gap (gemini 0.717 vs. claude 0.600). The identical ordering under disjoint tools, vocabulary, and constraints shows the difficulty axis is a property of the localization *task*, not of video editing.

The TSP recipe transfers untuned. We instantiate three checks from the pipeline spec—dependency-order (S2), per-operator cap (S3), cumulative-budget (S5)—and apply them *without tuning*; each localizes its target at $P/R = 1.00$ **with zero misfires** $(50/50)$. Composed with the gated no-fault probe (T0 $0.20 \rightarrow 0.60$), TSP reaches $\mathbf{0.883}$ from baseline 0.658 ($\Delta = +\mathbf{0.225}$), mirroring the video gain ($0.775 \rightarrow 0.872$). Once a constraint is a state predicate over the parsed log, localizing its violation reduces to an exact check in either domain.

Ecological validity: agent-authored pipelines. To rule out that template suites let the check exploit a fixed authoring style, we regenerate the domain-2 clean bases with an *LLM agent* (prompting CLAUDE-OPUS-4-8 to author varied legal pipelines, validated clean, then injecting the same faults with the same objective gold). All 60 agent-authored bases (10 per axis) passed with zero template fallbacks. Run untuned, the three checks again localize at $P/R = 1.00$, and full TSP lifts accuracy $0.725 \rightarrow \mathbf{0.842}$ (Figure 5). The ordering recurs on richer traces (S5 $0.90 \rightarrow 1.00$, S2 $0.70 \rightarrow 1.00$) and over-attribution is if anything *worse* (T0 baseline 0.00, gate 0.30)—the check is keyed on the constraint, not a template artifact.

6.2 Real-Trace Validation (P1-B)

On $1{,}617$ prelabeled runs from the $1{,}848$-run real pool, two frontier judges agree on failure presence only 68.2% of the time ($\kappa = 0.37$) and on the exact faulty step in only 66.5% of joint failure verdicts. Of 57 runs flagged despite a clean mechanical anchor, $53/54$ adjudicated cases are genuine semantic failures, predominantly off-goal drift (Appendix D). In a blind 60-task study, two humans reach

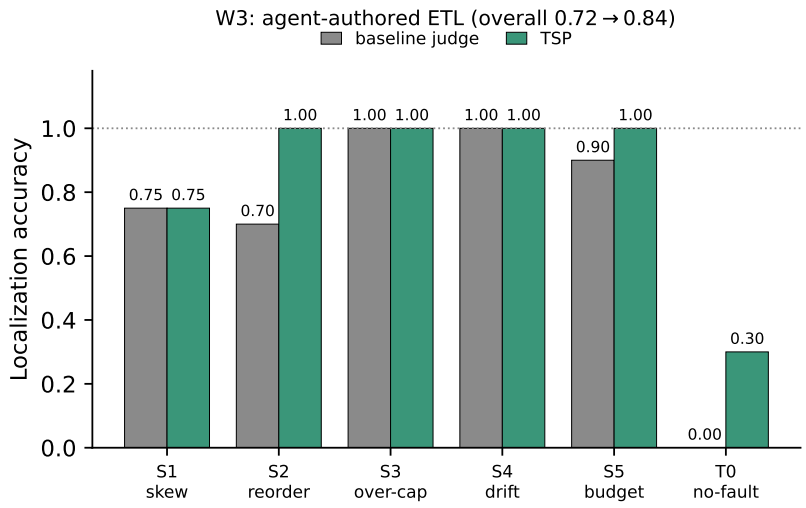


Figure 5: **Ecological validity.** Per-axis accuracy on 60 agent-authored ETL bases (varied predicates, transforms, schemas; 10 per axis), naive judge (baseline) vs. full TSP. The deterministic checks localize S2/S3/S5 at $P/R = 1.00$ untuned and the difficulty ordering recurs on these richer traces, confirming the effect is keyed on the constraint, not a template artifact.

$\kappa = 0.63$ (Landis & Koch, 1977), while the best automated judge reaches only 0.48 on consensus cases (Appendix D.1). Automated detection is therefore not reliable localization, even with multiple judges.

7 Discussion, Limitations, and Conclusion

Localization is distinct from task success: judges can solve local faults yet miss global constraints, or detect failure while inventing a culprit. TSP externalizes checkable trace-state computation before routing residual semantic judgments, lifting accuracy from 0.775 to 0.872. Its scope is explicit constraints; implicit goals still require a judge. The controlled track injects one fault per trace, real labels require targeted human review, and broader domains remain to be tested, although multi-generator and ETL results reduce single-template concerns. Appendix H details these boundaries.

AV-Fail makes earliest-fault localization measurable with exact controlled labels and $1{,}848$ real agent runs. It exposes a global-budget blind spot and clean-trace over-attribution that outcome metrics hide. Trace-State Prompting routes checkable state through deterministic tools and residual decisions through an LLM, improving localization from 0.775 to 0.872 and transferring to an ETL agent. The released data, injectors, and judge harness make this overlooked capability reproducible.

Ethics Statement

The controlled benchmark uses programmatically generated task specifications and synthetic fault injection, while the real-trace study uses task execution logs produced for this research. The released traces are screened to exclude personal information and credentials. Two annotators labeled a stratified sample under a fixed protocol; no sensitive attributes or high-stakes decisions are involved. We report disagreement and model limitations rather than treating automated judgments as ground truth.

Reproducibility Statement

The paper specifies the benchmark construction, fault taxonomy, evaluation metrics, model configurations, and statistical analyses. The appendix provides prompts, additional per-condition results, robustness checks, and annotation details. The supplementary archive contains the task data, raw results, analysis and figure-generation code, fixed seeds, dependency versions, and instructions for reproducing the reported tables and figures.

Use of Large Language Models

Large language models are central experimental subjects in this work: they generate agent traces, serve as failure-localization judges, and support the multi-model prelabeling and adjudication pipeline described in the paper. They were also used to assist retrieval, research execution, and drafting and polishing portions of the manuscript. All benchmark definitions, analyses, claims, citations, and final text were selected and verified by the authors.

References

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, et al. Why do multi-agent LLM systems fail? In *arXiv preprint arXiv:2503.13657*, 2025.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. Video-MME: The first-ever comprehensive evaluation benchmark of multi-modal LLMs in video analysis. 2024.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, et al. A survey on LLM-as-a-judge. In *arXiv preprint arXiv:2411.15594*, 2024.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. StableToolBench: Towards stable large-scale benchmarking on tool learning of large language models. In *Findings of the Association for Computational Linguistics (ACL Findings)*, 2024.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations (ICLR)*, 2024a.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, and Lichao Sun. MetaTool benchmark for large language models: Deciding whether to use tools and which to use. In *International Conference on Learning Representations (ICLR)*, 2024b.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023. doi: 10.1145/3571730.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations (ICLR)*, 2024.

J. Richard Landis and Gary G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, 1977. doi: 10.2307/2529310.

Haitao Li, Qingyao Dong, Junjie Chen, Huixue Su, Yujia Zhou, Qingyao Ai, Ziyi Ye, and Yiqun Liu. LLMs-as-judges: A comprehensive survey on LLM-based evaluation methods. In *arXiv preprint arXiv:2412.05579*, 2024.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-Bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *International Conference on Learning Representations (ICLR)*, 2024.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, et al. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations (ICLR)*, 2024.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. AgentBoard: An analytical evaluation board of multi-turn LLM agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Shahbaz Khan. Video-ChatGPT: Towards detailed video understanding via large vision and language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024a.

Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Shahbaz Khan. VideoGPT+: Integrating image and video encoders for enhanced video understanding. *arXiv preprint arXiv:2406.09418*, 2024b.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, et al. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general AI assistants. In *International Conference on Learning Representations (ICLR)*, 2024.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, et al. ToolLLM: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations (ICLR)*, 2024.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. In *arXiv preprint arXiv:2211.14275*, 2022.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6), 2024a.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024b.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024c.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, et al. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, et al. Which agent causes task failures and when? on automated failure attribution of LLM multi-agent systems. *arXiv preprint arXiv:2505.00212*, 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, et al. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, et al. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations (ICLR)*, 2024.

A Comparison to Existing Agent Benchmarks

Table 4 positions AV-FAIL against representative agent-evaluation benchmarks along the axes that matter for failure-step localization.

Benchmark family	Step-level	Neg. ctrl	Real track	Gold step
Task-success suites	×	×	✓	×
Process-supervision sets	✓	×	×	partial
LLM-as-judge quality	partial	×	✓	×
Failure attribution	✓	×	✓	partial
AV-FAIL (ours)	✓	✓	✓	✓

Table 4: AV-FAIL uniquely combines exact gold-step labels, a no-fault negative control, and a real-trace validation track. ("partial" step scores = scalar quality without exact-step attribution; the failure-attribution row covers Who&When (Zhang et al., 2025) and MAST (Cemri et al., 2025), which label naturally occurring failures without a programmatic gold step or a no-fault negative control.)

The comparison isolates the supervision gap addressed by AV-FAIL. Task-success suites retain realistic executions but do not identify the responsible step, whereas process-supervision sets expose intermediate judgments without a clean no-fault control or naturally occurring traces. Existing failure-attribution benchmarks come closest, but their partial labels cannot distinguish localization errors from annotation ambiguity. Combining programmatic gold steps with T0 controls and a separately validated real track lets us measure both missed faults and invented culprits under the same protocol.

B Fault Injector Details

Each injector is deterministic given a seed and modifies exactly one property of a valid trace (clip duration, clip order, transition length, subtitle offset, or the running duration total), recording the modified step as gold. T0 traces are passed through unchanged. Full injector code and per-condition manifests are released with the benchmark.

C Judge Models and Prompt

Judge models. The five-model controlled-track configuration uses Claude (`claude-opus-4-8`), Gemini (`gemini-3.5-flash`), Qwen (`qwen3.7-max`), DeepSeek (`deepseek-v4-pro`), and Doubao (`doubao-seed-2-0-pro`), queried in 2026-07 at temperature 0 with a 800-token cap (8000 for the reasoning model, whose thinking budget otherwise truncates the answer). The two mitigation experiments and the diverse-generator robustness check use the two instruction-following judges Claude and Gemini. Because these are hosted API models, exact weights may drift over time; the difficulty spectrum and the S5/T0 findings are the stable, reproducible signal (see the reproducibility note in the README).

Base prompt. The judge is given the task spec and the tool-call trace and asked to name the single step that introduced a spec violation, or -1 if the trace is fault-free. The system message states the role ("video-editing QA expert"), the tool semantics (`trim` with in_ms $<$ out_ms; `concat` by order; `transition` with a short `dur_ms`; `subtitle` whose span lies within its clip; `render`), and stresses that the constraints are *semantic* (pacing balance, transitions not too long, subtitles covering the opening, total duration within budget). The required output is {`"fault_idx":` `<int step idx, or -1 if none>`, `"reason":` `<one line>`}. We strip code fences, parse strictly, and exclude malformed outputs from the valid count.

Mitigation prefixes. Each mitigation prepends exactly one instruction to the base system message, holding everything else fixed. The *budget scaffold* instructs the judge to first sum each `trim` segment length plus every `transition` duration into a running total, compare it against

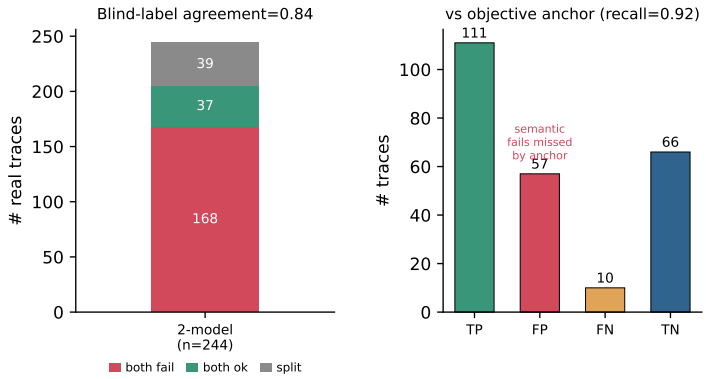


Figure 6: **Real-trace verification.** *Left:* inter-model agreement on has-failure over paired runs. *Right:* majority vote against the objective anchor; 57 false positives are semantic failures the anchor cannot see.

Real-track quantity	Count
Genuine agent runs (pool)	1,848
Prelabeled by two judges (scorable)	1,617
268-*run adjudication pilot:*	
Pilot prelabel set	268
routed to verification	122 (46%)
model split	39
model–anchor conflict	59
insufficient predictions	24
Adjudicated (valid verdict)	116
adjudicator agreed with both judges	82
adjudicator overrode both	11
adjudicator sided with anchor	5
Confirmed failures (has_failure)	92
off-goal drift	70
loop	18
tool error	4
Confirmed clean	24
Semantic failures missed by objective anchors	57

Table 5: Real-track (P1-B) verification funnel. The 57 semantic failures are runs the models flagged as failed but the objective anchor marked clean (`conflict_with_anchor`, model = True, anchor = False): overwhelmingly off-goal drift that mechanical detectors cannot see. Off-goal is the single most common confirmed failure kind (70 of 116).

`spec.max_total_ms`, and if it overflows, localize the `trim` step that first crosses (or finally breaks) the budget. The *no-fault gate* instructs the judge to first decide whether the trace truly violates the spec at all, and to emit a step index only when a clear violation is found, otherwise -1—so it does not invent a culprit merely because it was asked to find one.

D Real-Trace Verification Queue

The two judges prelabel all 1,617 scorable runs (has-failure agreement Cohen's $\kappa = 0.37$, raw 68.2%); the verification queue below comes from a 268-run adjudication pilot. Runs are routed to verification when the two judges split, when a judge conflicts with the objective anchor, or when predictions are insufficient. The adjudicator receives the full trace, both model verdicts, and the objective anchor, and emits {has_failure, fail_step, fail_kind, confidence, adjudication, reason}. The resulting queue (122 runs) is released to permit independent audit of the reference labels.

Judge (vs. human consensus, n=51)	agree	κ	over / miss
Human A vs. Human B (n=60)	0.85	**0.63**	—
CLAUDE-OPUS-4-8 (best)	0.78	0.48	3 / 8
DEEPSEEK-V3-2-251201 (cons.)	0.61	0.31	0 / 20
DOUBAO-SEED-2.0-PRO	0.69	−0.05	11 / 5

Table 6: Human ceiling on a blind 60-task set. Automated judges remain below human–human agreement and fail in opposite directions.

Fault type	Claude	DeepSeek	Qwen	All
S1 subtle-trim	1.00	0.64	1.00	0.73
S2 adjacent-swap	1.00	1.00	1.00	1.00
S3 transition-long	1.00	1.00	1.00	1.00
S4 subtitle-drift	1.00	1.00	1.00	1.00
S5 budget-creep	0.50	0.25	0.15	0.23
T0 no-fault	0.86	1.00	1.00	0.90

Table 7: Localization accuracy on 90 diverse-generator tasks, split by the agent that authored each trace (two-judge average). S5 remains the hardest regime and S1–S4 remain near-ceiling for all three generators, showing the difficulty spectrum is not an artifact of the trace generator.

D.1 Human-Ceiling Breakdown

We estimate the attainable agreement level on the real-track judgment task using a blind, stratified set of 60 traces independently labeled by two humans. The automated comparison is restricted to the 51 cases on which the annotators reach consensus, preventing human disagreement from being counted as a model error. Table 6 reports raw agreement, chance-corrected κ, and the direction of binary errors relative to that consensus.

Human–human agreement is substantial ($\kappa = 0.63$), while the strongest automatic judge reaches only 0.48. More importantly, the automatic judges do not share one calibratable failure mode: Claude misses more failures than it invents, DeepSeek is strongly conservative, and Doubao over-attributes faults. This directional disagreement supports the main-text conclusion that model ensembles cannot substitute for targeted human review on ambiguous real traces.

E Difficulty Spectrum Across Trace Generators

To test whether our findings depend on the identity of the trace-synthesizing agent, we generate 90 additional tasks with three different planning agents (Claude, DeepSeek, Qwen; 30 each, balanced across the six fault types) and score them with two judges (Claude, Gemini). Table 7 reports localization accuracy per fault type and generator. The ordering is stable: the mechanical faults S1–S4 stay near ceiling for every generator, while the cumulative budget fault S5 collapses regardless of author (0.50/0.25/0.15), and clean traces (T0) again invite over-attribution. This mirrors the main-set spectrum on the same two judges (S5= 0.31, T0-FP= 0.35), confirming the effects are judge-side properties of the video-editing task rather than a generator artifact.

F General-Reasoning Baselines (B4–B6)

Table 8 reports the full general-reasoning comparison summarized as Finding 4 in the main text: three off-the-shelf test-time-reasoning methods—self-consistency (B4), self-refine (B5), and step-wise PRM-style scoring (B6)—against the same two instruction-following judges (Claude, Gemini) used for B1, alongside the targeted/routed approaches (B2, B3, TSP). None of the general methods closes the S5 budget-creep blind spot (≤ 0.26 vs. the 0.31 B1 baseline), and B6 significantly lowers overall accuracy—confirming the gap is a mechanism deficit, not a compute deficit.

Method	S5	T0	Overall	vs. B1
B1 holistic judge	0.310	0.740	0.775	—
General reasoning methods				
B4 self-consistency	0.260	0.830	0.792	$n.s.$
B5 self-refine	0.170	0.940	0.742	$\downarrow^{*}$
B6 stepwise PRM	0.240	0.980	0.665	$\downarrow^{***}$
Targeted / routed				
B2 no-fault gate	0.270	**0.910**	0.767	$n.s.$
B3 budget scaffold	0.820	0.580	0.848	$\uparrow^{***}$
TSP (ours)	**0.890**	**0.910**	**0.872**	$\uparrow^{***}$

Table 8: Exact-step accuracy of general reasoning methods versus targeted and routed approaches on the two hard regimes, all on the *same* two instruction-following judges (Claude, Gemini) for an apples-to-apples paired comparison. General test-time reasoning (self-consistency B4, self-refine B5, stepwise PRM B6) leaves the S5 budget-creep blind spot essentially unmoved (≤ 0.26, versus the 0.31 B1 baseline) and never matches the routed TSP; B6 even hurts overall ($p = 1.4\times10^{-7}$, paired McNemar vs. B1). Arrows/$n.s.$ mark paired-McNemar significance vs. B1 ($^{*}p<0.05$, $^{***}p<10^{-4}$).

Fault axis	**Baseline**	**TSP**
S1 skew (partition)	0.80	0.80
S2 reorder (stage)	0.75	**1.00**
S3 over-cap (memory)	1.00	**1.00**
S4 schema drift	0.90	0.90
S5 budget creep	**0.30**	**1.00**
T0 no-fault	**0.20**	0.60
Overall	0.658	**0.883**

Table 9: **Cross-domain replication** (data-pipeline agent, 300 tasks, claude+gemini): the video-domain ordering recurs and untuned TSP recovers the same gain ($\Delta = +0.225$).

G Cross-Domain (ETL) Agent-Authored Bases

On the cross-domain (ETL) track we first replicate the difficulty spectrum on a matched 300-task template suite (50 per axis; data-pipeline agent, judged by claude+gemini). Table 9 shows the video-domain ordering recurs and untuned TSP recovers the same gain ($0.658 \rightarrow 0.883$, $\Delta = +0.225$).

To further rule out that template suites let the check exploit a fixed authoring style, we regenerate the ETL clean bases with a planning agent: 60 agent-authored data-pipeline bases (10 per axis; varied predicates, transforms, aggregations, and schemas), validated clean, then injected with the same faults and objective gold. The three deterministic checks again localize at $P/R = 1.00$ untuned and TSP lifts overall accuracy $0.725 \rightarrow 0.842$, mirroring the video-editing ordering (S5 $0.90 \rightarrow 1.00$, S2 $0.70 \rightarrow 1.00$) while over-attribution is if anything worse (T0 baseline 0.00, gate 0.30); see Figure 5 in the main paper for the per-axis breakdown.

H Extended Discussion

This appendix expands the scope, cost, and limitations summarized at the end of the main-paper Discussion.

When can trace state be computed? TSP's determinism is also its scope. Stage 1 fires only where a constraint is a checkable *predicate* over the parsed trace against a *declared* spec—a budget ceiling, an ordering requirement, or a per-step bound—as on the controlled and ETL tracks, where the task ships a machine-readable target (`max_total_ms`, a required stage order, a numeric cap). Under these conditions the check is not a heuristic but a recomputation of the exact quantity the fault violates, which is why it localizes S2/S3/S5 at precision/recall 1.00 *untuned* across two domains. On the real track the situation inverts: a free-form user request ("make it punchier," "fix the timing") carries no machine-readable ceiling, and the dominant failure kind is *off-goal* drift (Appendix D)—a trajectory/intent mismatch that no fixed predicate captures because the "spec" lives only in natural

language. TSP therefore does *not* subsume real-track judging: Stage 1 degrades to a no-op and the gated Stage 2 model judge decides alone, which is exactly why the real-track numbers show no deterministic lift. The recipe is thus narrower than the slogan "compute the state": *whatever part of a constraint is machine-checkable should be checked, not prompted*, with only the residual semantic judgment routed to the model. This suggests a concrete research program: automatically *extracting* checkable predicates from natural instructions (e.g., parsing "keep it under a minute" into a duration bound, or "don't add captions" into a forbidden-op set) would widen Stage 1's reach on the real track. Our harness already exposes the hook where such an extractor would plug in, between trace parsing and the Stage 1 check battery.

The cost of restraint. The no-fault gate reduces false alarms but suppresses some true detections (visible as the S4 dip)—a detection/precision tension fundamental to *any* monitor that can also abstain, and measurable here only because AV-FAIL includes a negative control (clean traces with no injected fault). Without such a control the over-attribution mode is invisible: a judge that flags 0.35 of clean traces looks indistinguishable from one that is merely aggressive on true faults, and a single accuracy number rewards the former. Because the gate is a tunable threshold, deployments can pick an operating point along this curve—conservative (high precision, misses some faults) for low-stakes triage, or sensitive (high recall, more false alarms) for safety-critical oversight where a missed failure is costlier than a false one. Our protocol makes this choice explicit rather than baking in one default, and we report both operating regimes so downstream users can locate their own point.

I EXTENDED RELATED WORK

Trajectory and process evaluation. Process-supervision and step-level reward models score intermediate reasoning (Lightman et al., 2024; Uesato et al., 2022; Wang et al., 2024c), and "LLM-as-judge" methods assess trajectories holistically (Zheng et al., 2023; Li et al., 2024; Gu et al., 2024), though such judges carry systematic biases (Wang et al., 2024b). We differ in requiring *exact-step* attribution against gold labels rather than a scalar score, and in constructing faults whose localization demands *global* reasoning over the whole trace (e.g. a running duration budget) that step-local scorers are not designed to catch. A complementary line equips models with external computation or verifiers over intermediate steps (Schick et al., 2023; Yao et al., 2023; Lightman et al., 2024; Uesato et al., 2022); our Trace-State Prompting (Section 5) brings this to the *judge*, computing the checkable trace state deterministically and consulting the model only for the residual decision.

Failure analysis and self-correction. Work on self-refinement (Madaan et al., 2023), reflection (Shinn et al., 2023), and error taxonomy (Cemri et al., 2025) studies recovery once a failure is noticed, yet Huang et al. (2024a) show self-correction is unreliable. Localization is the missing precondition: the hardest cases (budget drift) are those a judge fails to register, while clean traces provoke over-attribution—reminiscent of hallucination (Ji et al., 2023).

Video-editing agents. Automated video editing has advanced rapidly (Maaz et al., 2024b), atop video-language models (Maaz et al., 2024a) benchmarked mainly on comprehension (Fu et al., 2024); agentic editing is likewise evaluated on output quality. We repurpose it because correctness hinges on ordering, timing, and global budgets—properties yielding controllable fault types unavailable in single-answer tasks. To our knowledge AV-FAIL is the first to treat failure-step localization as a standalone, reproducible benchmark with fault injectors, a judge harness, and a real-trace verification protocol.

J TSP CHECK ROBUSTNESS AND HELD-OUT FAULTS

The check is robust, not tuned. The one free parameter, the 15% tolerance, sits on a stable plateau: sweeping $5\% \to 15\%$ holds precision at 1.00 (triggers $34 \to 40$), degrading only at $\geq 20\%$ (0.87). Applied *unchanged* to the 90 multi-generator traces of Section 4.1 (three different planners), Stage 1 fires 14 times at 100% precision and 0.93 S5 recall with *zero* misfires—a property of the constraint, not of the trace set it was designed on.

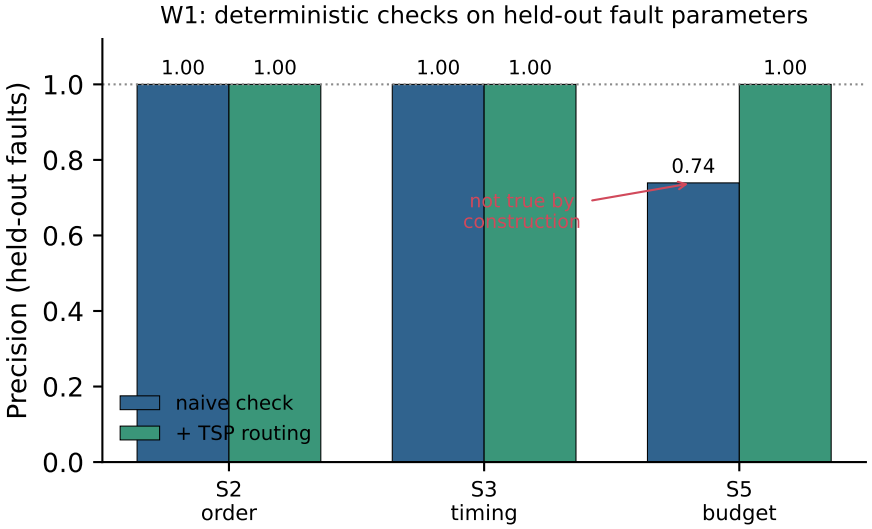


Figure 7: Held-out fault generalization. With parameters *disjoint* from each predicate, the order (S2) and timing (S3) checks retain $P = R = 1.00$; the budget check (S5) drops to naive precision 0.739 because 96 over-transition (S3) traces also overflow the timeline. TSP's Stage-1 timing precedence pre-empts them, restoring budget-check precision to 1.000 at recall 0.962.

Held-out faults: not true by construction. Because the checks are validated on faults injected with the parameters they key on, $P = R = 1.00$ could be tautological. We therefore re-inject each fault with parameters *disjoint from* the check predicate—non-adjacent swaps/reversals (S2), durations in $[850, 3000]$ ms (S3), overruns in $[50, 1500]$ ms with trim spreads at the 13% edge of the frozen 15% tolerance (S5)—and apply the predicates unchanged. The order and timing checks retain $P = R = 1.00$ on all 300 held-out traces, confirming they encode the constraint, not a memorised value. The budget check is where held-out injection bites: in isolation it drops to precision 0.739 (recall 0.962), because 96 held-out over-transition (S3) traces are *also* over budget, so the arithmetic check sees a real overrun but blames a trim step. This overlapping-fault ambiguity is exactly what motivates *routing rather than stacking*: consulting the higher-precision timing check first (Stage-1 precedence) pre-empts those traces and restores budget-check precision to 1.000 at recall 0.962 (Figure 7), leaving 10 residual misses (3.8%) as an honest limitation.

K Computational Resources and API Usage

No local GPU training. All experiments use hosted API endpoints; no local GPU training was performed. TSP (Trace-State Prompting) checks are deterministic string/arithmetic operations on CPU.

API models. Five frontier models were used as agents and judges via an OpenAI-compatible gateway: CLAUDE-OPUS-4.8, GEMINI-3.5-FLASH, DEEPSEEK-V4-PRO, QWEN3.7-MAX, and DOUBAO-SEED-2.0-PRO-260215. Agent generation temperature 0.3; judge temperature 0.0. The six-model Fleiss κ study added GPT-5.5 as a sixth blind annotator.

Total API calls.

- **Controlled track:** 300 tasks $\times$ 5 models (baseline) $+$ 100 tasks $\times$ 2 models (mitigation) $\approx$ **1,900 calls**.
- **Real track:** $1{,}848$ traces $\times$ 5 models $\approx$ **9,240 calls**.
- **Real track prelabel:** 268 traces $\times$ 6 models $=$ **1,608 calls**.
- **Adjudication:** 122 traces $\times$ 2 annotators $+$ 122 adjudicator $\approx$ **366 calls**.

Total: approximately **13,100 API calls** across all experiments.

Cost estimate. At 2026-07 hosted-API pricing, the total cost was approximately **$200–300 USD**.

Hosted API drift. As these are hosted API versions available at run time (2026-07), exact weights may drift over time and absolute numbers may shift. However, the reported *effects*—the S5 blind spot and T0 over-attribution failure modes, and the relative difficulty ordering—are stable.