RETAC: RESIDUAL TEST-TIME AGENTIC COMPUTATION FOR VIDEO SPATIAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time agentic computation can improve multimodal large language models (MLLMs) through planning, tool use, iterative perception, and reflection, but indiscriminate computation may duplicate resolved work, introduce noisy evidence, and perturb correct predictions. We propose that such computation should be *residual*: allocate additional computation only to requirements unresolved by the base model and, after a localized failure, recompute only the affected states. We instantiate this formulation in **ReTAC**, a training-free multiagent framework for video spatial reasoning. ReTAC organizes supplementary computation around unresolved requirements. It plans ordered steps to acquire missing spatial evidence and combines the observations with compatible native evidence for reasoning. In addition, it uses stage-aligned verification to localize failures to planning, perception, or reasoning. Dependency-based recovery recomputes only affected states while reusing unaffected ones. Extensive experiments show that ReTAC achieves the highest overall score among the compared methods across three mainstream MLLMs and two widely adopted benchmarks. Further analyses show that gains from additional computation depend not only on how much is used, but also on where it is allocated and which valid intermediate states are preserved.

1 INTRODUCTION

Multimodal large language models (MLLMs) are increasingly augmented at test time with agentic computation, such as planning, external tools, iterative perception, and reflection, aiming to solve problems that native inference alone cannot resolve well. Video spatial reasoning makes the allocation of such computation particularly challenging because answer-relevant evidence is often distributed across frames and heterogeneous spatial cues. Solving a query often requires tracking object identities over time, accounting for camera motion, recovering depth and geometry, and aligning observations across spatial reference frames. These capabilities are relevant to embodied agents, AR/VR, and autonomous systems (Yang et al., 2025b; Dongfang et al., 2026; Li et al., 2026c). Existing methods improve access to spatial evidence through auxiliary representations, including depth maps, camera poses, cognitive maps, and structured 3D representations (Qi et al., 2026; Fan et al., 2026; Cheng et al., 2026; Hua et al., 2026; Zhang et al., 2026; Ruan et al., 2026), or through external tools that acquire additional evidence at inference time (Zhang et al., 2026; Li et al., 2026a; Wang et al., 2026; Dai et al., 2026). Additional evidence, however, is not necessarily beneficial: ViSRA reports that providing Qwen2.5-VL with a ground-truth cognitive map can reduce performance (Mou et al., 2026). This raises a more specific question: *given what the base model has already resolved, what additional test-time computation is still needed?*

The answer depends on both the model and the instance. For the same video and question, one MLLM may resolve all required evidence natively, another may lack only a particular spatial relation, and a third may require additional perception. Adaptive agents condition tool use on task context, scene state, intermediate evidence, feedback, or acquired skills (Li et al., 2026a; Wang et al., 2026). Such adaptation does not explicitly distinguish requirements already supported by the current model from those that remain unresolved. Supplementary processing can therefore reacquire supported evidence, introduce noisy observations, and perturb an otherwise correct prediction. We formulate *residual test-time agentic computation*: supplementary computation targets only requirements unresolved by native inference, while compatible native evidence is retained.

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.

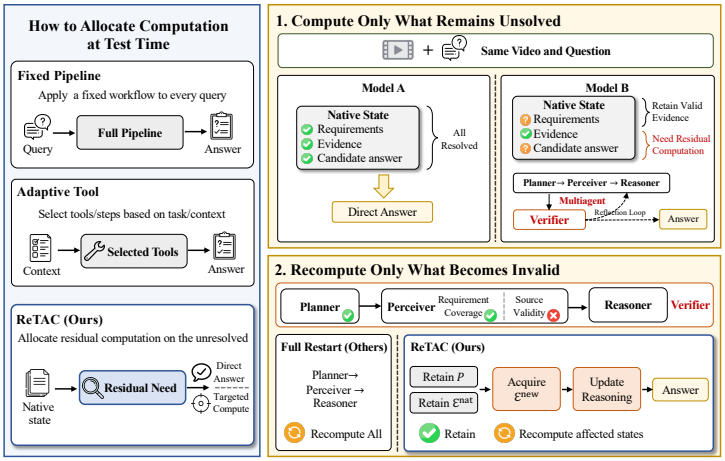

Figure 1: **Residual test-time agentic computation. Left:** fixed, context-adaptive, and residual computation allocation. **Top right:** model–instance routing from the native state to direct return or targeted refinement. **Bottom right:** dependency-aware selective recovery after a localized failure, while retaining valid evidence.

Residual computation also applies to recovery. Agentic inference is multistage: planning may omit a required condition, perception may produce a source-inconsistent observation, and reasoning may misuse valid evidence. Existing systems use reflection or consistency checks to assess evidence sufficiency, frame quality, or object consistency (Mou et al., 2026; Jiang et al., 2026). However, when a failure is localized, restarting the entire pipeline discards valid intermediate state and repeats unaffected computation. Residual recovery instead recomputes only the invalidated state and its dependent states. Fig. 1 summarizes these two principles: *compute only what remains unresolved, and recompute only what becomes invalid.*

We instantiate these principles in **ReTAC** (**Re**sidual **T**est-Time **A**gentic **C**omputation), a training-free multiagent framework for video spatial reasoning. ReTAC first produces a structured native state containing task requirements, source-linked native evidence, unresolved requirements, and a candidate answer. If no unresolved requirement is reported, ReTAC returns the native candidate directly. Otherwise, residual planning constructs supplementary perception steps only for the unresolved requirements. Targeted spatial perception executes these steps, and the resulting evidence is integrated with native evidence for reasoning. ReTAC then applies stage-aligned verification to planning, perception, and reasoning. A localized failure triggers replanning, reperception, or rereasoning, while ReTAC preserves and reuses unaffected intermediate states. ReTAC therefore uses the current residual state to determine what additional computation and recovery are needed, rather than repeatedly executing a complete agentic pipeline.

We evaluate ReTAC on VSI-Bench-Debiased and STI-Bench with Qwen3-VL-8B, Gemini 3.1 Pro, and GPT-5.5. ReTAC achieves the highest overall score among the compared methods for all three models on both benchmarks. Controlled routing interventions directly test computation allocation: forcing supplementary processing on instances selected for direct return lowers performance, while bypassing supplementary processing on instances selected for residual refinement also degrades performance for all three models. At the default recovery budget, selective recovery uses fewer mean model calls than full restart for all three models, with higher overall scores for Qwen and GPT and only a slight 0.08-point decrease for Gemini. These results indicate that test-time scaling is allocation-sensitive: its utility depends not only on the amount of computation, but also on where computation is applied and which valid states are preserved.

Our main contributions are threefold. First, we formulate **residual test-time agentic computation**, which allocates supplementary computation to requirements unresolved by the current model–instance pair and restricts recovery to invalidated states and their dependencies. Second, we introduce **ReTAC**, a training-free multiagent framework that realizes this formulation through structured native-state generation, residual routing and planning, targeted spatial perception, stage-aligned verification, and dependency-based selective recovery. Third, across three heterogeneous MLLMs and two video spatial reasoning benchmarks, ReTAC achieves the highest overall score among the compared methods in all six model–benchmark settings. Ablation studies and further analyses support its two core designs: residual allocation targets computation to unresolved requirements, while selective recovery reduces model calls relative to full restart with comparable overall performance.

2 Related Work

Video Spatial Reasoning with MLLMs. VSI-Bench and STI-Bench reveal persistent limitations of MLLMs in video spatial reasoning (Yang et al., 2025a; Li et al., 2025b). Existing methods mainly provide the model with richer spatial evidence through explicit scene representations or additional evidence gathering. Explicit representations take several forms: GPT4Scene uses bird's-eye views and consistent object markers to link global and local views (Qi et al., 2026), TRACE records scene context, camera trajectories, and object entities as structured text (Hua et al., 2026), and AlloSpatial combines geometric and semantic evidence in query-conditioned allocentric representations (Ruan et al., 2026). Additional spatial evidence can be obtained through Think3D's 3D reconstruction and iterative viewpoint changes (Zhang et al., 2026) or ViSRA's specialized spatial tools for multi-role inference (Mou et al., 2026). These approaches improve how spatial evidence is represented or acquired. ReTAC instead treats supplementary spatial processing as an on-demand mechanism, invoking it only when native evidence is insufficient.

Adaptive Test-Time Agentic Computation. Adaptive multimodal agents decide when and how to invoke tools during inference (Wang et al., 2026; Song et al., 2026). AdaTooler-V learns selective invocation from tool-benefit rewards (Wang et al., 2026), while AdaReasoner selects tools across multiple steps (Song et al., 2026). DART recruits tools from agent disagreement, and ToolScope coordinates global planning with local visual perception (Sivakumaran et al., 2026; Deng et al., 2026). Agent-X further evaluates stepwise reasoning and tool use in vision-centric tasks, revealing persistent failures in multimodal reasoning chains (Ashraf et al., 2026). For spatial reasoning, Skill-3D retrieves skills relevant to the scene and task for iterative evidence acquisition (Li et al., 2026a). Their allocation signals include task context, intermediate outcomes, disagreement, and prior tool-use experience. ReTAC bases this decision on the residual state of the current model–instance pair, distinguishing supported native evidence from unresolved requirements.

Multimodal Reflection and Selective Recovery. Reflection-based methods revisit visual evidence or intermediate reasoning after an initial attempt. Recent methods encourage explicit self-reflection or evaluate intermediate reasoning steps with multimodal process supervision (Wang et al., 2025a;c). VISCO studies visual critique and correction, showing that self-generated feedback can be unreliable (Wu et al., 2025). GThinker learns cue-guided visual rethinking (Zhan et al., 2026), while REVISOR returns to video segments relevant to the question (Li et al., 2026b). Refer-Agent uses reflection to refine frame selection and object analysis (Jiang et al., 2026), and ViSRA separates planning, reflection, execution, and answer synthesis across specialized roles (Mou et al., 2026). These methods improve evidence re-examination and feedback during reflection. ReTAC determines recovery scope after a localized failure: stage-aligned verification identifies whether the error lies in planning, perception, or reasoning, and dependency tracking limits recomputation to invalidated states while preserving unaffected states.

3 Method

3.1 Native State and Residual Routing

Given a video V, a spatial reasoning question q, and optional answer choices $\mathcal{O}$, ReTAC predicts an answer with a base MLLM M and an operation interface $\mathcal{T}$. The same model serves as Planner, Perceiver, Reasoner, and Verifier under role-specific instructions, without task-specific training. ReTAC makes two allocation decisions: whether native inference requires supplementary computation and, after a localized verification failure, which states must be recomputed. Fig. 2 gives an overview.

ReTAC begins with a single native call that receives only $x = (V, q, \mathcal{O})$, without tool descriptions or outputs, and produces the following state:

$$S^{\mathrm{nat}} = \mathrm{Native}_M(x) = \big(\mathcal{R}^{\mathrm{nat}}, \mathcal{E}^{\mathrm{nat}}, \mathcal{U}^{\mathrm{nat}}, \hat{y}^{\mathrm{nat}}\big), \tag{1}$$

where $\mathcal{R}^{\mathrm{nat}}$ contains the identified task requirements, $\mathcal{E}^{\mathrm{nat}}$ contains source-linked native evidence, $\mathcal{U}^{\mathrm{nat}}$ contains unresolved requirements, and $\hat{y}^{\mathrm{nat}}$ is the native candidate answer. Each requirement is labeled `SUPPORTED` or `UNRESOLVED`, and supported requirements reference records in $\mathcal{E}^{\mathrm{nat}}$.

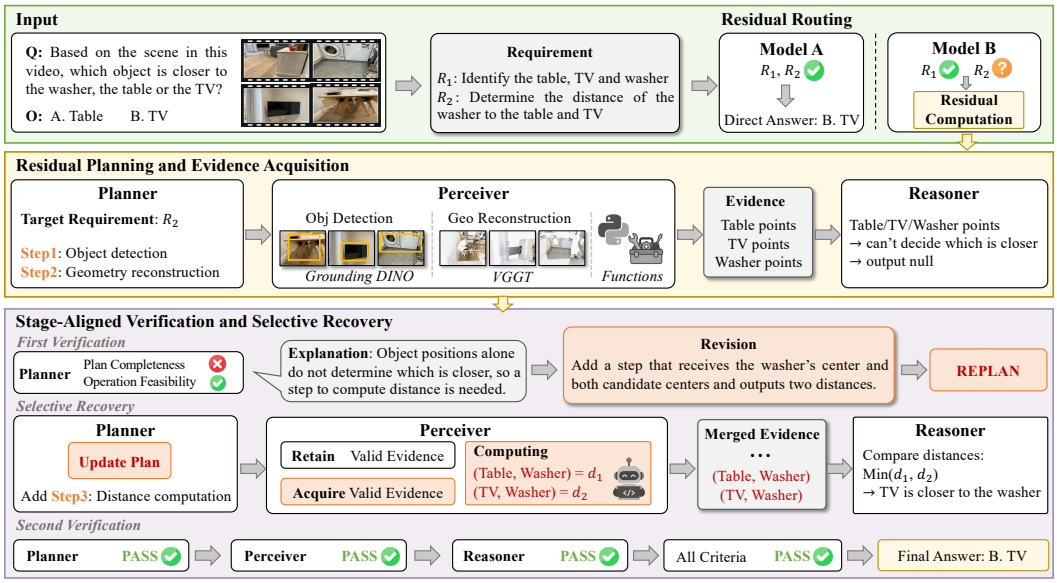

Figure 2: **Overview of ReTAC.** Native inference either returns the candidate when all identified requirements are supported or routes unresolved requirements to residual refinement. The illustrated refinement path detects an incomplete plan. `REPLAN` adds only the missing distance computation while retaining valid evidence.

The requirements specify the information needed to answer the question, such as object identities, spatial relations, and temporal conditions. For example, native inference may identify the relevant objects but leave their spatial relation unresolved. This decomposition lets subsequent computation retain supported observations and acquire only the missing evidence. ReTAC routes the instance as

$$\hat{y} = \begin{cases} \hat{y}^{\text{nat}}, & \mathcal{U}^{\text{nat}} = \varnothing, \\ \text{Refine}_M(x, S^{\text{nat}}, \mathcal{T}), & \text{otherwise}, \end{cases} \tag{2}$$

where Refine_M comprises residual planning, perception, reasoning, and verification with selective recovery. The first branch returns `DIRECT` as a routing decision rather than a verification result.

3.2 Residual Planning and Evidence Acquisition

Residual Planning. For an instance routed to refinement, the Planner extends the native state only where additional information is needed. It constructs $P_0 = (\mathcal{S}_0, \mathcal{R}_0, \mathcal{U}_0)$, where $\mathcal{S}_0$ is an ordered sequence of supplementary steps. Requirements omitted by native inference can be added as $\mathcal{R}_0^{\text{add}}$:

$$\mathcal{R}_0 = \mathcal{R}^{\text{nat}} \cup \mathcal{R}_0^{\text{add}}, \qquad \mathcal{U}_0 = \mathcal{U}^{\text{nat}} \cup \mathcal{R}_0^{\text{add}}. \tag{3}$$

Each step targets a requirement and invokes either a single primitive operation or a composition of primitives. For each primitive operation, the plan specifies the arguments, input references, dependencies, and expected output. The Planner constructs each composition according to the requirements of the task. Inputs may reference the video, native evidence, or preceding operation outputs. A step is included only if it resolves a requirement in $\mathcal{U}_0$ or supplies a prerequisite for doing so. The interface $\mathcal{T}$ provides GroundingDINO for object detection (Liu et al., 2024), VGGT for geometry estimation (Wang et al., 2025b), and predefined spatial functions. Appendix C describes these tools and functions in detail.

Targeted Perception. The Perceiver executes $\mathcal{S}_0$ in dependency order using the declared operations and arguments. Each observation is stored as:

$$e_j = (r_j, o_j, \sigma_j, p_j, \mathcal{D}_j), \tag{4}$$

where r_j is the target requirement, o_j the observed content, σ_j the sources supporting that content, p_j the step producing the observation ($p_j = \text{nat}$ for native observations), and $\mathcal{D}_j$ the input on which

Table 1: **VSI-Bench-Debiased results.** Parentheses report ReTAC minus the corresponding method within the same base model. Count, Abs.D, Obj.S, Room, Rel.Dir, Rel.D, Order, and Route denote object counting, absolute distance, object size, room size, relative direction, relative distance, appearance order, and route planning. S&T denotes See&Trek (Li et al., 2025a).

Base	Method	Numerical (MRA, %)				Multiple Choice (Acc., %)				Avg.
		Count	Abs.D	Obj.S	Room	Rel.Dir	Rel.D	Order	Route	
Qwen3 VL 8B	Direct	44.82 (+8.37)	39.75 (+3.80)	59.24 (+2.37)	58.60 (-1.30)	41.67 (+13.59)	49.35 (+5.49)	60.38 (+8.49)	28.07 (+5.26)	47.73 (+5.76)
	CoT	45.12 (+8.07)	39.63 (+3.92)	59.15 (+2.46)	58.32 (-1.02)	43.23 (+12.03)	50.96 (+3.88)	61.95 (+6.92)	30.41 (+2.92)	48.59 (+4.90)
	S&T	43.55 (+9.64)	39.98 (+3.57)	**62.04** (-0.43)	55.55 (+1.75)	37.20 (+18.06)	**55.16** (-0.32)	66.35 (+2.52)	26.32 (+7.01)	48.27 (+5.22)
	Think3D	44.86 (+8.33)	39.82 (+3.73)	60.14 (+1.47)	**58.75** (-1.45)	42.56 (+12.70)	52.37 (+2.47)	61.85 (+7.02)	31.58 (+1.75)	48.99 (+4.50)
	ReTAC	**53.19**	**43.55**	61.61	57.30	**55.26**	54.84	**68.87**	**33.33**	**53.49**
Gemini 3.1 Pro	Direct	41.12 (+7.76)	31.31 (+3.62)	56.26 (+2.38)	46.20 (+6.05)	46.44 (+7.35)	59.03 (+7.10)	64.15 (+2.52)	45.61 (+3.51)	48.77 (+5.03)
	CoT	42.36 (+6.52)	31.57 (+3.36)	57.12 (+1.52)	45.98 (+6.27)	48.96 (+4.83)	62.15 (+3.98)	66.46 (+0.21)	47.37 (+1.75)	50.24 (+3.56)
	S&T	44.26 (+4.62)	30.20 (+4.73)	**59.49** (-0.85)	44.10 (+8.15)	50.44 (+3.35)	**68.39** (-2.26)	**69.70** (-3.03)	48.53 (+0.59)	51.89 (+1.91)
	Think3D	43.37 (+5.51)	30.32 (+4.61)	57.23 (+1.41)	47.33 (+4.92)	52.65 (+1.14)	63.76 (+2.37)	68.24 (-1.57)	47.37 (+1.75)	51.28 (+2.52)
	ReTAC	**48.88**	**34.93**	58.64	**52.25**	**53.79**	66.13	66.67	**49.12**	**53.80**
GPT-5.5	Direct	49.56 (+0.40)	**45.51** (-0.58)	63.88 (+1.83)	57.55 (+0.85)	64.92 (+4.64)	67.42 (+0.75)	71.70 (+1.88)	67.54 (+1.18)	61.01 (+1.37)
	CoT	50.23 (-0.27)	44.57 (+0.36)	63.82 (+1.89)	57.87 (+0.53)	65.66 (+3.90)	67.96 (+0.21)	72.64 (+0.94)	68.42 (+0.30)	61.40 (+0.98)
	S&T	**50.60** (-0.64)	45.07 (-0.14)	62.63 (+3.08)	53.00 (+5.40)	66.69 (+2.87)	65.27 (+2.90)	70.75 (+2.83)	65.79 (+2.93)	59.98 (+2.40)
	Think3D	48.79 (+1.17)	43.28 (+1.65)	63.77 (+1.94)	57.73 (+0.67)	67.45 (+2.11)	**68.71** (-0.54)	72.43 (+1.15)	67.84 (+0.88)	61.26 (+1.12)
	ReTAC	49.96	44.93	**65.71**	**58.40**	**69.56**	68.17	**73.58**	**68.72**	**62.38**

the observation depends. Each record preserves the object identity, temporal reference, coordinate frame, and scale required to interpret o_j. Source references σ_j point to sampled frames or raw tool outputs, allowing later verification to distinguish errors in evidence acquisition from errors in the Perceiver's interpretation. Failed acquisitions are logged without creating an observation.

Supplementary observations $\mathcal{E}^{\text{new}}$ are merged with $\mathcal{E}^{\text{nat}}$ to form the active evidence set $\mathcal{E}_0$ for reasoning. Conflicting observations remain in $\mathcal{E}_0$ with provenance and dependencies for verification. Raw outputs are retained in the execution history independently of observations derived from them.

Evidence-Grounded Reasoning. The Reasoner computes $(Z_0, \hat{y}_0) = \text{Reason}_M(q, O, P_0, \mathcal{E}_0)$, where Z_0 is the reasoning trace. Each critical claim must reference active evidence or a preceding supported claim. The Reasoner introduces no new visual observations. The resulting state $\Sigma_0 = (P_0, \mathcal{E}_0, Z_0, \hat{y}_0)$ remains provisional until verification.

3.3 Stage-Aligned Verification

To localize errors before recovery, the Verifier checks the current state $\Sigma_t = (P_t, \mathcal{E}_t, Z_t, \hat{y}_t)$ in the order of planning, perception, and reasoning, where t denotes the number of completed recovery actions and $t = 0$ the initial refinement state. The base model evaluates semantic support, while deterministic checks validate interfaces, references, and traceable computations.

Planning. Given $(q, \mathcal{O}, P_t, \mathcal{T})$, the Verifier checks *plan completeness* and *operation feasibility*: whether the plan includes steps needed to address every requirement not supported by native evidence and whether declared operations can produce the required evidence. To isolate planning errors, this stage does not access the video, observations, reasoning trace, or candidate answer. An omission is considered necessary if the missing information could change the answer while all represented requirements are held fixed. Feasibility is assessed from documented operation interfaces.

Perception. The Verifier checks *source validity* and *requirement coverage* against the cited frames and raw outputs. Native and supplementary observations follow the same criteria: each active record must be supported by its source, and requirements in $\mathcal{R}_t$ must be resolved by valid active evidence.

Reasoning. The Verifier checks *evidence support*, *inference validity*, and *answer consistency* along $\mathcal{E}_t \to Z_t \to \hat{y}_t$. These checks validate the premises, the resulting semantic or computational derivations, and the mapping from the final conclusion to the returned answer, including option mapping or numerical normalization when required.

Each criterion returns a verdict $v_{t,c} \in \{\texttt{PASS}, \texttt{UNKNOWN}, \texttt{FAIL}\}$, with a target and justification. For stage $a \in \{\text{plan}, \text{perc}, \text{reason}\}$ with criteria $\mathcal{C}_a$, the stage verdict is:

$$v_t^a = \min_{c \in \mathcal{C}_a} v_{t,c}, \qquad \texttt{FAIL} \prec \texttt{UNKNOWN} \prec \texttt{PASS}. \tag{5}$$

Table 2: **STI-Bench results.** Parentheses report ReTAC minus the corresponding method within the same base model. DM, SR, VG, DPL, SA, EO, TD, and PE denote dimensional measurement, spatial relation, 3D video grounding, displacement and path length, speed and acceleration, egocentric orientation, trajectory description, and pose estimation. S&T denotes See&Trek (Li et al., 2025a).

Base	Method	Static Understanding			Dynamic Understanding					Avg.
		DM	SR	VG	DPL	SA	EO	TD	PE	
Qwen3 VL 8B	Direct	29.76 (-0.23)	47.26 (+4.79)	39.75 (+4.73)	19.83 (+9.50)	38.37 (+3.62)	57.84 (+5.94)	39.74 (+20.52)	57.22 (+0.28)	39.87 (+4.63)
	CoT	30.10 (-0.57)	48.63 (+3.42)	40.06 (+4.42)	23.46 (+5.87)	39.88 (+2.11)	58.92 (+4.86)	46.15 (+14.11)	56.94 (+0.56)	41.23 (+3.27)
	S&T	29.76 (-0.23)	47.03 (+5.02)	37.85 (+6.63)	23.18 (+6.15)	37.16 (+4.83)	62.16 (+1.62)	44.45 (+15.81)	55.83 (+1.67)	40.28 (+4.22)
	Think3D	**34.95** (-5.42)	50.00 (+2.05)	38.17 (+6.31)	28.59 (+0.74)	35.04 (+6.95)	56.76 (+7.02)	42.31 (+17.95)	53.42 (+4.08)	40.87 (+3.63)
	ReTAC	29.53	**52.05**	**44.48**	**29.33**	**41.99**	**63.78**	**60.26**	**57.50**	**44.50**
Gemini 3.1 Pro	Direct	40.95 (+2.65)	61.64 (-0.68)	43.11 (+0.11)	33.24 (+4.47)	41.79 (+2.42)	63.06 (+0.18)	61.11 (+8.12)	70.65 (+0.18)	49.47 (+1.86)
	CoT	40.48 (+3.12)	**62.33** (-1.37)	42.90 (+0.32)	34.92 (+2.79)	42.60 (+1.61)	63.78 (-0.54)	65.38 (+3.85)	70.28 (+0.55)	50.00 (+1.33)
	S&T	40.60 (+3.00)	59.81 (+1.15)	40.59 (+2.63)	34.82 (+2.89)	42.90 (+1.31)	**66.85** (-3.61)	59.40 (+9.83)	69.17 (+1.66)	49.37 (+1.96)
	Think3D	42.56 (+1.04)	62.10 (-1.14)	40.80 (+2.42)	36.59 (+1.12)	42.90 (+1.31)	62.35 (+0.89)	67.95 (+1.28)	69.26 (+1.57)	50.08 (+1.25)
	ReTAC	**43.60**	60.96	**43.22**	**37.71**	**44.21**	63.24	**69.23**	**70.83**	**51.33**
GPT-5.5	Direct	36.33 (+5.19)	69.18 (-1.14)	41.64 (+3.15)	37.71 (+6.98)	45.32 (+1.31)	51.35 (-0.36)	69.23 (+1.71)	70.00 (-0.37)	49.61 (+2.53)
	CoT	37.02 (+4.50)	**69.86** (-1.82)	41.32 (+3.47)	39.66 (+5.03)	46.22 (+0.41)	51.89 (-0.90)	70.51 (+0.43)	69.72 (-0.09)	50.24 (+1.90)
	S&T	37.60 (+3.92)	65.75 (+2.29)	39.96 (+4.83)	38.55 (+6.14)	**47.63** (-1.00)	**53.15** (-2.16)	**71.36** (-0.42)	**70.19** (-0.56)	50.08 (+2.06)
	Think3D	39.67 (+1.85)	68.72 (-0.68)	39.86 (+4.93)	42.83 (+1.86)	46.32 (+0.31)	50.45 (+0.54)	70.51 (+0.43)	69.82 (-0.19)	50.76 (+1.38)
	ReTAC	**41.52**	68.04	**44.79**	**44.69**	46.63	50.99	70.94	69.63	**52.14**

The stage passes only if all criteria pass. A single `FAIL` makes the stage fail and any `UNKNOWN` makes it non-passing. Verification stops at the first non-passing stage and returns a localized diagnosis with a revision target. If a later check raises a concrete concern about an earlier pass, only the implicated upstream criterion is rechecked before recovery is selected. Appendix A.2 specifies the criteria and stage-specific inputs.

3.4 Selective Recovery

Recovery Scope. The controller selects a recovery action only when verification supports a concrete revision. `REPLAN` revises requirements or acquisition steps, including operations, arguments, inputs, or dependencies. `REPERCEIVE` repeats an affected acquisition under the unchanged plan. If the acquisition procedure itself must change, the controller instead selects `REPLAN`. `REREASON` preserves the plan and evidence and regenerates the derivation and answer. Each recovery call receives the correction target, diagnosis, proposed revision, and accumulated recovery history.

Dependency-Based Reuse. Let Δ_t denote the records or steps directly changed or invalidated by recovery. The affected set is $\mathcal{I}_t = \Delta_t \cup \mathrm{Desc}(\Delta_t)$, where Desc contains all transitive dependents under the recorded input and support references. Replanning retains evidence compatible with the revised plan, reperception preserves unrelated evidence, and rereasoning reuses the plan and all active evidence. Superseded records remain in the execution history but are removed from the active evidence set. Cached checks are reused only when their supporting records and justifications remain valid. Replacing one observation, for example, invalidates its source check and any coverage or reasoning checks that depend on it, while checks on unrelated observations can be retained. Every recovery regenerates $(Z_{t+1}, \hat{y}_{t+1})$ as a single unit, while unaffected plan steps, evidence, and verification checks are reused.

Reverification and Termination. The revised state is verified after each recovery. ReTAC returns `VERIFIED` only when planning, perception, and reasoning all pass; the status denotes acceptance under the specified criteria rather than ground-truth correctness. The budget T counts recovery actions, with verification performed at $t = 0$ and after every permitted action, including the last. If no supported revision is available or the budget is exhausted before acceptance, ReTAC returns the current answer with status `UNVERIFIED`. Algorithm 1 in Appendix A.4 gives the full procedure.

4 Experiments

4.1 Experimental Setup

We evaluate ReTAC with Qwen3-VL-8B (Bai et al., 2025), Gemini 3.1 Pro, and GPT-5.5 on VSI-Bench-Debiased (Yang et al., 2025a; Brown et al., 2025) and STI-Bench (Li et al., 2025b). VSI-

Table 3: **Component replacements.** Parentheses report each variant minus ReTAC in percentage points. Gen. Plan. and Gen. Refl. denote General Planning and General Reflection, respectively.

Base	Method	Numerical (MRA, %)				Multiple Choice (Acc., %)				Avg.
		Count	Abs.D	Obj.S	Room	Rel.Dir	Rel.D	Order	Route	
Qwen3 VL 8B	**ReTAC**	**53.19**	**43.55**	**61.61**	57.30	**55.26**	**54.84**	**68.87**	**33.33**	**53.49**
	Gen. Plan.	51.69 (-1.50)	41.16 (-2.39)	60.89 (-0.72)	**57.92** (+0.62)	52.58 (-2.68)	49.78 (-5.06)	65.41 (-3.46)	31.58 (-1.75)	51.37 **(-2.12)**
	Gen. Refl.	52.43 (-0.76)	42.39 (-1.16)	61.54 (-0.07)	56.13 (-1.17)	53.23 (-2.03)	52.36 (-2.48)	64.78 (-4.09)	29.82 (-3.51)	51.58 **(-1.91)**
GPT-5.5	**ReTAC**	**49.96**	**44.93**	**65.71**	**58.40**	**69.56**	**68.17**	**73.58**	**68.72**	**62.38**
	Gen. Plan.	49.53 (-0.43)	44.24 (-0.69)	64.13 (-1.58)	58.20 (-0.20)	67.92 (-1.64)	67.10 (-1.07)	72.33 (-1.25)	66.67 (-2.05)	61.27 **(-1.11)**
	Gen. Refl.	49.18 (-0.78)	44.21 (-0.72)	65.38 (-0.33)	57.03 (-1.37)	68.27 (-1.29)	65.81 (-2.36)	72.01 (-1.57)	64.91 (-3.81)	60.85 **(-1.53)**

Bench-Debiased uses mean relative accuracy (MRA) for numerical tasks and accuracy for multiple-choice tasks, averaging eight task scores uniformly. STI-Bench uses accuracy for all eight tasks and a sample-weighted overall score. We report means over three runs for the main experiments, component ablations, and recovery cost and budget analyses. The routing and answer-transition analyses use a single run for each base model. We uniformly sample 32 frames per video and set the default recovery budget $T = 2$. All tool-enabled variants share the same operation interface. The same base MLLM instantiates the Planner, Perceiver, Reasoner, and Verifier under role-specific instructions shared across models. Appendix B provides full evaluation and intervention protocols.

4.2 Main Results

We compare ReTAC with Direct QA, chain-of-thought prompting (CoT), See&Trek (Li et al., 2025a), and Think3D (Zhang et al., 2026). ReTAC obtains the highest score among the compared methods in all six model–benchmark settings (Tables 1 and 2).

On VSI-Bench-Debiased, the clearest gains occur on tasks that require spatial evidence across frames or viewpoints. Relative direction improves for all three models, while Qwen and Gemini also improve on counting and appearance order. These categories align with the object localization, temporal evidence, and observer-frame computation targeted by residual refinement. Absolute distance and room size are less consistent across models, which may partly reflect VGGT's lack of metric scale. This task-wise pattern links the gains to the evidence available through the tools.

On STI-Bench, scores for displacement and path length improve substantially for all three base models, while trajectory description shows the largest gains for Qwen and Gemini. Both tasks require spatial evidence to be accumulated over time, matching ReTAC's targeted acquisition of unresolved temporal and geometric cues. The gains vary across tasks and models: several categories regress slightly, and GPT gains less than Gemini on VSI-Bench-Debiased but more on STI-Bench. This reversal suggests that the benefit of residual refinement depends more on the unresolved requirements of a model–instance pair and the evidence exposed by the tools than on base model strength alone.

4.3 Residual Computation Allocation

All subsequent analyses use VSI-Bench-Debiased. Component replacements are evaluated with Qwen and GPT, while routing and recovery analyses use all three models.

Native-State Conditioning. To test the role of native-state conditioning, we replace residual routing and planning with a general adaptive planner that receives the input, task requirements, and tool interfaces, but not native support labels or native evidence. It can return the stored native answer or acquire evidence for the full task. Performance decreases for both models, with lower scores in 15 of 16 task-level comparisons (Table 3). These results indicate that, under a limited budget, focusing computation on the unresolved is more effective than reacquiring evidence for the full task.

Routing Interventions. For each model, we keep ReTAC's original direct return (S_1) and residual refinement (S_2) subsets fixed. On S_1, we force supplementary acquisition for every native requirement while retaining native evidence. On S_2, we suppress refinement and return the native candidate. Subset scores are computed by averaging per-question scores within each fixed routing subset. Both forced alternatives reduce subset performance for every model (Fig. 3). The larger penalty changes across base models: Qwen and Gemini lose more performance when refinement is suppressed,

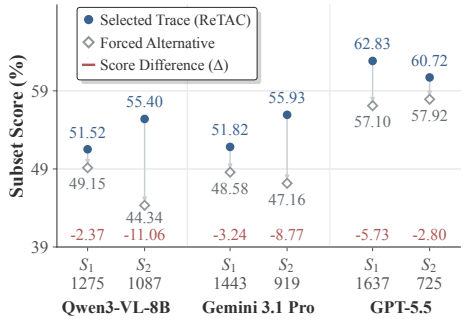

Figure 3: **Routing interventions on fixed subsets.** S_1 and S_2 are direct-return and refinement subsets. Circles show selected-route scores, diamonds the forced alternative, and Δ labels the within-subset change.

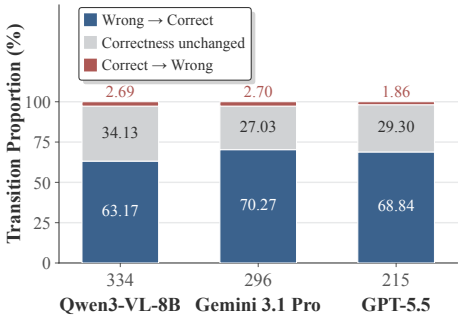

Figure 4: **Answer transitions after recovery.** Bars partition recovered instances into wrong-to-correct, unchanged correctness, and correct-to-wrong. X-axis labels show the number of recovered instances.

whereas GPT suffers a larger drop when supplementary acquisition is forced. This paired result supports model–instance-specific computation allocation rather than a uniformly deeper inference.

4.4 Residual Recovery

Structured Verification and Localized Revision. Replacing stage-aligned verification and selective recovery with general reflection and full restart lowers every task score for both evaluated models (Table 3). The reflector receives the same trajectory and sources, so the comparison concerns how errors are localized and revisions are organized. The degradation suggests that effective reflection depends on localizing the failure before revision, rather than having the verifier assess the entire inference trajectory at once.

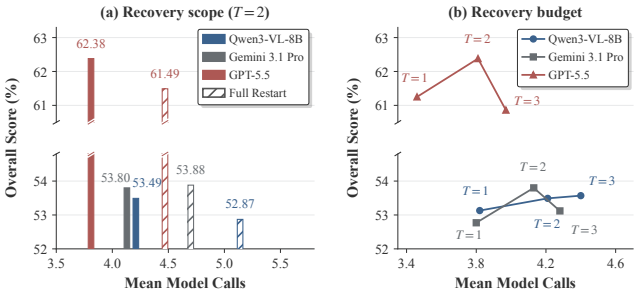

Figure 5: **Recovery scope and budget.** (a) Selective recovery versus full restart at $T = 2$. (b) Score–cost trade-offs as T increases. Beyond $T = 2$, calls increase with little or negative score gains.

Answer Transitions after Recovery. We compare the initial supplementary answer $\hat{y}_0$ with the final answer only for instances that trigger at least one recovery action. For this transition analysis, numerical answers are considered correct if their relative error with respect to the ground truth is at most 25%. Recovery is invoked for roughly 30% of each model's refinement subset. Among these instances, about two-thirds of transitions are wrong-to-correct, whereas correct-to-wrong transitions remain below 3% for all three models (Fig. 4). These transitions indicate that revisions usually repair an incorrect initial answer while rarely overturning a correct answer.

4.5 Recovery Cost and Budget

Selective Recovery versus Full Restart. To isolate recovery scope, both strategies start from the same state and use identical stage-aligned verification with $T = 2$. Full restart reruns the entire refinement trajectory from the fixed native state, including planning, perception, and reasoning. Selective recovery recomputes only invalidated states and their dependencies. Selective recovery reduces mean model calls for all three models (Fig. 5; counting rule in Appendix B.1). Qwen and GPT improve in both cost and score, whereas Gemini exhibits a small cost–accuracy trade-off.

Recovery Budget. As shown in Fig. 5, increasing T from 1 to 2 improves the overall score for every base model. Raising the budget to 3 increases model calls for all three, but yields only a small gain for Qwen and lower scores for Gemini and GPT. Thus, a larger recovery budget does not consistently improve performance.

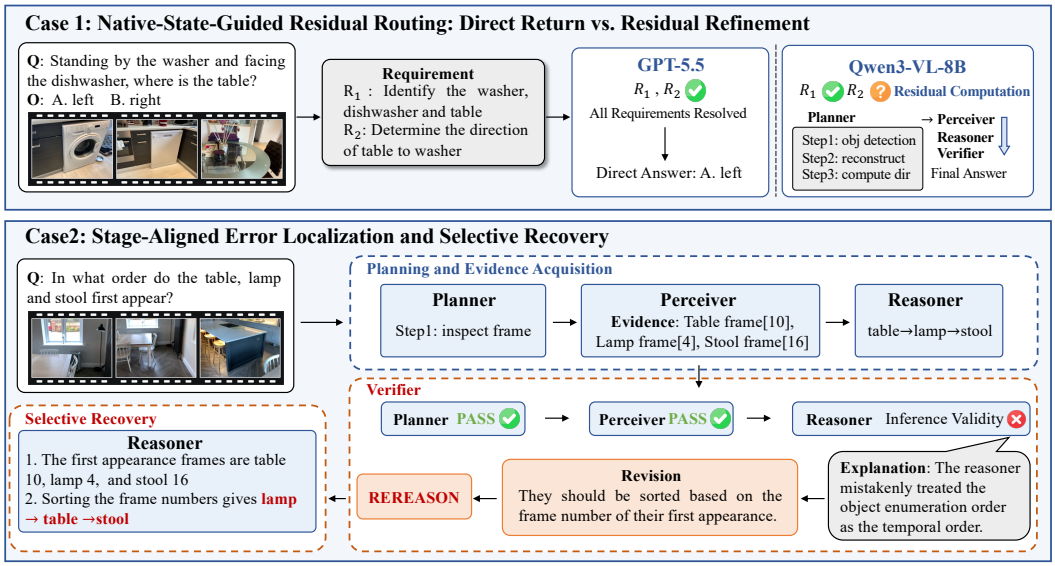

Figure 6: **Qualitative examples of residual allocation and recovery. Top:** The same query follows different routes because GPT-5.5 resolves both native requirements, whereas Qwen3-VL-8B leaves the directional relation unresolved. **Bottom:** Verification localizes an appearance-order error to reasoning. REREASON corrects the order while reusing the verified plan and evidence.

4.6 Qualitative Analysis

Fig. 6 shows the two residual decisions: compute only for unresolved requirements, then recompute only invalidated states and their dependents.

Case 1: Native-State-Guided Residual Routing. The query requires identifying the washer, dishwasher, and table (R_1) and resolving the table's direction in the specified observer frame (R_2). GPT-5.5 supports both requirements and returns its native candidate, *A: left*. Qwen3-VL-8B supports R_1 but leaves R_2 unresolved, so refinement adds detection, reconstruction, and direction computation. The supported identity evidence remains active rather than being reacquired. The same query follows different execution paths across base models, illustrating that residual computation is determined by the current model–instance state rather than by the task alone.

Case 2: Stage-Aligned Error Localization and Selective Recovery. For the appearance-order query, the Perceiver records first appearances at frames 10, 4, and 16 for the table, lamp, and stool, but the Reasoner preserves the enumeration order and returns *table* $\rightarrow$ *lamp* $\rightarrow$ *stool*. Planning and perception pass verification, while the inference-validity check localizes the error to reasoning. Guided by the Verifier's revision to sort objects by their first-appearance frames, ReTAC applies REREASON without changing P_t or $\mathcal{E}_t$. The retained frame indices, $4 < 10 < 16$, establish the corrected order *lamp* $\rightarrow$ *table* $\rightarrow$ *stool* without new perception. This correction updates only the reasoning trace and answer, illustrating dependency-based state reuse.

5 Conclusion

We introduced ReTAC, a training-free multiagent framework that formulates test-time agentic computation for video spatial reasoning as residual computation: supplementary computation targets only requirements unresolved by native inference, while recovery recomputes only invalidated states and their dependencies. Across three MLLMs and two benchmarks, ReTAC achieves the highest overall score among the compared methods. Controlled routing interventions show that forcing refinement on direct-return instances or suppressing refinement on refinement instances reduces performance. Selective recovery reduces mean model calls relative to full restart with comparable overall performance. The results support model–instance specific computation allocation and dependency-based state reuse over repeatedly executing a full agentic pipeline.

Statement on AI Use

We have not used generative AI tools to generate the core research ideas or design the proposed method. Tasks involving the formulation and proof of theoretical claims are not applicable to this work. For tasks with recommended disclosure, we used generative AI tools to polish the manuscript's language, including correcting grammar, refining wording, and improving clarity and readability. We have reviewed all AI-assisted work and revised the suggested edits to ensure that they preserved the intended technical meaning and accurately reflected the research. We take full responsibility for the accuracy, originality, and integrity of the final manuscript.

References

Tajamul Ashraf, Amal Saqib, Hanan Gani, Muhra AlMahri, Yuhao Li, Noor Ahsan, Umair Nawaz, Jean Lahoud, Hisham Cholakkal, Mubarak Shah, et al. Agent-x: Evaluating deep multimodal reasoning in vision-centric agentic tasks. In *International Conference on Learning Representations*, volume 2026, pp. 49814–49874, 2026.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Tal Dimry, Yuri Feigin, Peter Fu, Thomas Gebauer, Brandon Joffe, Daniel Kurz, Arik Schwartz, et al. Arkitscenes: A diverse real-world dataset for 3d indoor scene understanding using mobile rgb-d data. *arXiv preprint arXiv:2111.08897*, 2021.

Ellis Brown, Jihan Yang, Shusheng Yang, Rob Fergus, and Saining Xie. Benchmark designers should "train on the test set" to expose exploitable non-visual shortcuts, 2025. URL https://arxiv.org/abs/2511.04655.

An-Chieh Cheng, Yang Fu, Yatai Ji, Ligeng Zhu, Guanqi Zhan, Zhuoyang Zhang, Zhaojing Yang, Song Han, Yao Lu, Pavlo Molchanov, et al. Grounded 3d-aware spatial vision-language modeling. *arXiv preprint arXiv:2605.30307*, 2026.

Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 2432–2443. IEEE, 2017.

Yalun Dai, Hao Li, Shulin Tian, Runmao Yao, Yuhao Dong, Fangzhou Hong, Zhaoxi Chen, Fangfu Liu, Baoliang Tian, Dingwen Zhang, et al. S-agent: Spatial tool-use elicits reasoning for spatial intelligence. *arXiv preprint arXiv:2606.20515*, 2026.

Mengjie Deng, Guanting Dong, and Zhicheng Dou. Toolscope: An agentic framework for vision-guided and long-horizon tool use. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 211–225, 2026.

Zihao Dongfang, Xu Zheng, Ziqiao Weng, Yuanhuiyi Lyu, Danda Pani Paudel, Luc Van Gool, Kailun Yang, and Xuming Hu. Are multimodal large language models ready for omnidirectional spatial reasoning? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, pp. 9759–9769, June 2026.

Zhiwen Fan, Jian Zhang, Renjie Li, Junge Zhang, Runjin Chen, Hezhen Hu, Kevin Wang, Peihao Wang, Huaizhi Qu, Shijie Zhou, et al. Vlm-3r: Vision-language models augmented with instruction-aligned 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 31054–31065, 2026.

Jiacheng Hua, Yishu Yin, Yuhang Wu, Tai Wang, Yifei Huang, and Miao Liu. Unleashing spatial reasoning in multimodal large language models via textual representation guided reasoning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13616–13637, 2026.

Haichao Jiang, Tianming Liang, Wei-Shi Zheng, and Jian-Fang Hu. Refer-agent: A collaborative multi-agent system with reasoning and reflection for referring video object segmentation. *arXiv preprint arXiv:2602.03595*, 2026.

Haoyuan Li, Zhengdong Hu, Jun Wang, Hehe Fan, and Yi Yang. Skill-3d: Evolving scene-aware skills for agentic 3d spatial reasoning. *arXiv preprint arXiv:2606.07436*, 2026a.

Jiaze Li, Hao Yin, Wenhui Tan, Jingyang Chen, Boshen Xu, Yuxun Qu, Yijing Chen, Jianzhong Ju, Zhenbo Luo, and Jian Luan. Revisor: Beyond textual reflection, towards multimodal introspective reasoning in long-form video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5059–5069, 2026b.

Pengteng Li, Pinhao Song, Wuyang Li, Huizai Yao, Weiyu Guo, Yijie Xu, Dugang Liu, and Hui Xiong. See&trek: Training-free spatial prompting for multimodal large language model. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=2exr4mlbx1.

Yifei Li, Pengyiang Liu, Yuhang Zang, Zhongyue Shi, Qi Fu, Hongye Hao, and Jiwen Lu. Ovo-s-bench: A hierarchical benchmark for streaming spatial intelligence in multimodal llms. *arXiv preprint arXiv:2606.03890*, 2026c.

Yun Li, Yiming Zhang, Tao Lin, XiangRui Liu, Wenxiao Cai, Zheng Liu, and Bo Zhao. Sti-bench: Are mllms ready for precise spatial-temporal world understanding? In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5622–5632. IEEE, 2025b.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European conference on computer vision*, pp. 38–55. Springer, 2024.

Tingshu Mou, Jiabo He, Renying Wang, Ce Liu, Hao Yang, Tiehua Zhang, Jingjing Chen, and Xingjun Ma. Visra: A video-based spatial reasoning agent for multi-modal large language models. *arXiv preprint arXiv:2605.10106*, 2026.

Zhangyang Qi, Zhixiong Zhang, Ye Fang, Jiaqi Wang, and Hengshuang Zhao. Gpt4scene: Understand 3d scenes from videos with vision-language models. In *International Conference on Learning Representations*, volume 2026, pp. 157277–157302, 2026.

Shouwei Ruan, Bin Wang, Zhenyu Wu, Qihui Zhu, Yuxiang Zhang, Jingzhi Li, Yubin Wang, and Xingxing Wei. Allospatial: Agentic harness framework for spatial reasoning in foundation models. *arXiv preprint arXiv:2606.08952*, 2026.

Nithin Sivakumaran, Justin Chen, David Wan, Yue Zhang, Jaehong Yoon, Elias Stengel-Eskin, and Mohit Bansal. Dart: Leveraging multi-agent disagreement for tool recruitment in multimodal reasoning. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5445–5464, 2026.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, volume 2025, pp. 10131–10165, 2025.

Mingyang Song, Haoyu Sun, Jiawei Gu, Linjie Li, Ranjay Krishna, and Yu Cheng. Adareasoner: Dynamic tool orchestration for iterative visual reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 142731–142754, 2026.

Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2443–2451. IEEE, 2020.

Pengzhan Sun, Ramanathan Rajaraman, Shiu-Hong Kao, Junbin Xiao, and Angela Yao. Resti: A source-grounded audit and repair of sti-bench, 2026. URL https://arxiv.org/abs/2609.24727.

Chaoyang Wang, Kaituo Feng, Dongyang Chen, Zhongyu Wang, Zhixun Li, Sicheng Gao, Meng Meng, Xu Zhou, Manyuan Zhang, Yuzhang Shang, et al. Adatooler-v: Adaptive tool-use for images and videos. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 18058–18072, 2026.

Haozhe Wang, Chao Qu, Zuming Huang, Wei Chu, Fangzhen Lin, and Wenhu Chen. Vl-rethinker: Incentivizing self-reflection of vision-language models with reinforcement learning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 30865–30891. Curran Associates, Inc., 2025a. doi: 10.52202/085713-1037. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/2c84844a559e4f962752570bff456ae4-Paper-Conference.pdf.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5294–5306. IEEE, 2025b.

Weiyun Wang, Zhangwei Gao, Lianjie Chen, Zhe Chen, Jinguo Zhu, Xiangyu Zhao, Yangzhou Liu, Yue Cao, Shenglong Ye, Xizhou Zhu, et al. Visualprm: An effective process reward model for multimodal reasoning. *arXiv preprint arXiv:2503.10291*, 2025c.

Xueqing Wu, Yuheng Ding, Bingxuan Li, Pan Lu, Da Yin, Kai-Wei Chang, and Nanyun Peng. Visco: Benchmarking fine-grained critique and correction towards self-improvement in visual reasoning. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9527–9537. IEEE, 2025.

Jihan Yang, Shusheng Yang, Anjali W Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. Thinking in space: How multimodal large language models see, remember, and recall spaces. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10632–10643. IEEE, 2025a.

Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, et al. Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. *arXiv preprint arXiv:2502.09560*, 2025b.

Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12–22. IEEE, 2023.

Yufei Zhan, Ziheng Wu, Yousong Zhu, Rongkun Xue, Guanghao Zhou, Ruipu Luo, Zhenghao Chen, Can Zhang, Yifan Li, Zhentao He, et al. Gthinker: Towards general multimodal reasoning via cue-guided rethinking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11954–11965, 2026.

Jiyao Zhang, Weiyao Huang, Bo Peng, Mingdong Wu, Fei Hu, Zijian Chen, Bo Zhao, and Hao Dong. Omni6dpose: A benchmark and model for universal 6d object pose estimation and tracking. In *European Conference on Computer Vision*, pp. 199–216. Springer, 2024.

Zaibin Zhang, Yuhan Wu, Lianjie Jia, Yifan Wang, Zhongbo Zhang, Yijiang Li, Binghao Ran, Fuxi Zhang, Zhuohan Sun, Zhenfei Yin, et al. Think3d: Thinking with space for spatial reasoning. *arXiv preprint arXiv:2601.13029*, 2026.

SUPPLEMENTARY MATERIAL

ReTAC: Residual Test-Time Agentic Computation for Video Spatial Reasoning

Anonymous Authors

A RESIDUAL INFERENCE, STAGE-ALIGNED VERIFICATION, AND SELECTIVE RECOVERY

This appendix specifies the state semantics, stage-specific verification inputs, and recovery rules underlying Sections 3.1–3.4. We retain the notation of the main text and focus on the details needed to reproduce residual routing, verification, and dependency-aware state reuse.

A.1 STATE REPRESENTATION AND EVIDENCE INTEGRATION

Native State. The native call in Eq. 1 receives only $x = (V, q, \mathcal{O})$, without the operation interface or tool outputs. Each $r \in \mathcal{R}^{\mathrm{nat}}$ is labeled `SUPPORTED` or `UNRESOLVED`, and the unresolved set is

$$\mathcal{U}^{\mathrm{nat}} = \{r \in \mathcal{R}^{\mathrm{nat}} : \mathrm{status}(r) = \texttt{UNRESOLVED}\}. \tag{6}$$

These labels determine routing; they are not verification outcomes. Because the Planner is invoked only after an instance is routed to refinement, requirements omitted by the native call can be introduced only on that branch.

Residual Plan. For $\mathcal{U}^{\mathrm{nat}} \neq \varnothing$, the Planner conditions on the native state and the documented interface:

$$P_0 = (\mathcal{S}_0, \mathcal{R}_0, \mathcal{U}_0) = \mathrm{Plan}_M(x, S^{\mathrm{nat}}, \mathcal{T}). \tag{7}$$

It may add requirements omitted by native inference, with $\mathcal{R}_0$ and $\mathcal{U}_0$ defined by Eq. 3. Each step $s_k \in \mathcal{S}_0 = (s_1, \ldots, s_K)$ specifies a target requirement, a primitive operation or a composition of primitive operations, arguments, input references, dependencies, and expected output. Operations follow dependency order both within and across steps. Steps are included only when they resolve an element of $\mathcal{U}_0$ or provide a prerequisite for doing so. Thus, $\mathcal{U}_0$ records acquisition targets at planning time rather than verified resolutions. The number of steps K is determined by the Planner and has no fixed upper bound.

Evidence Records and Integration. Native and supplementary observations share the record structure in Eq. 4. The source field σ_j identifies the frame indices, raw operation record, or record field supporting o_j, whereas $\mathcal{D}_j$ records the inputs used to produce the observation. For an observation produced by s_k, $r_j = \mathrm{req}(s_k)$ and $p_j = s_k$; native observations use $p_j = \mathrm{nat}$.

The Perceiver resolves the declared inputs and executes each step without altering its operation, arguments, or acquisition scope. Failed acquisitions are logged without creating an observation. If $\mathcal{E}_k^{\mathrm{new}}$ denotes the records produced by s_k, then

$$\mathcal{E}^{\mathrm{new}} = \mathrm{Perceive}_M(x, P_0, \mathcal{T}) = \bigcup_{k=1}^{K} \mathcal{E}_k^{\mathrm{new}}, \tag{8}$$

and the initial active evidence set is

$$\mathcal{E}_0 = \mathrm{Merge}(\mathcal{E}^{\mathrm{nat}}, \mathcal{E}^{\mathrm{new}}). \tag{9}$$

The merge operation does not collapse conflicting observations: each record remains associated with its source and dependencies. Raw outputs are retained in the execution history even when a derived observation is superseded. During recovery, the corrected record replaces its predecessor only in the active evidence set.

Evidence-Grounded Reasoning. The Reasoner uses $(q, O, P_0, \mathcal{E}_0)$ to produce $(Z_0, \hat{y}_0)$. Every critical claim in Z_0 must reference active evidence or a preceding supported claim. The Reasoner cannot introduce new visual observations. Any additional visual observation must be acquired by the Perceiver and registered with its source in the active evidence set before the Reasoner can use it. The plan specifies what information must be resolved but does not itself constitute evidence.

A.2 Stage-Aligned Verification Details

For $\Sigma_t = (P_t, \mathcal{E}_t, Z_t, \hat{y}_t)$, verification proceeds in the order planning $\rightarrow$ perception $\rightarrow$ reasoning. The model evaluates semantic support, while deterministic checks validate interfaces, references, and traceable computations.

Outcomes and Aggregation. Each criterion returns `PASS`, `FAIL`, or `UNKNOWN` and includes a target and justification. Equation 5 gives the stage aggregation rule: a stage passes only if every criterion passes, and `UNKNOWN` is non-passing. Verification stops at the first non-passing stage, so later stages are unevaluated rather than assigned `UNKNOWN`.

Planning. The planning check receives only $(q, \mathcal{O}, P_t, \mathcal{T})$; it does not access the video, observation contents, reasoning trace, or candidate answer. Deterministic checks use declared metadata for references and dependencies.

Plan completeness. The plan must cover the entities, relations, and spatial or temporal conditions needed to answer the question. A missing acquisition is necessary when the information it would provide could change the answer while the represented requirements are held fixed. An identifiable omission yields `FAIL`; insufficient information to determine completeness yields `UNKNOWN`.

Operation feasibility. Deterministic checks validate registered primitives, required arguments, types and ranges, input references, and dependencies both within composite steps and across plan steps. The semantic check assesses whether the documented outputs and limitations of the selected operations can support their target requirements. Invalid interfaces or incompatible acquisition procedures yield `FAIL`; indeterminate feasibility yields `UNKNOWN`. A pass establishes feasibility, not correctness of the acquired observation.

Perception. This stage receives P_t, $\mathcal{E}_t$, the cited frames, raw operation outputs, and the relevant interface definitions. Native and supplementary records are evaluated by the same criteria.

Source validity. Reference checks validate frame indices, operation records, and cited output fields. The semantic check then determines whether the cited source supports the observation under the specified object identity and spatial or temporal context. A definite inconsistency yields `FAIL`; inaccessible or ambiguous support yields `UNKNOWN`.

Requirement coverage. Every $r \in \mathcal{R}_t$ must be resolved by valid active evidence. Missing or contradicted evidence yields `FAIL`; unresolved support or relevance yields `UNKNOWN`. Neither the number of acquired records nor the Perceiver's execution status determines coverage.

Reasoning. This stage receives $(q, \mathcal{O}, P_t, \mathcal{E}_t, Z_t, \hat{y}_t)$ and checks the derivation from active evidence to the returned answer.

Evidence support. Reference checks require every critical claim to point to active evidence or preceding supported claims; semantic checks verify that those sources support the stated premises. Invalid references, contradicted premises, or identifiable unsupported assumptions yield `FAIL`; ambiguous support yields `UNKNOWN`.

Inference validity. Deterministic checks are used when the operators and operands are traceable to active evidence; other semantic or geometric inferences are evaluated by the model. A definite logical or computational error yields `FAIL`; an inference that cannot be verified or rejected yields `UNKNOWN`.

Answer consistency. The returned answer must agree with the conclusion in Z_t. For multiple-choice questions, labels or exact option text are mapped to option identifiers. Numerical answers are normalized under the required unit conversion and rounding rules. Disagreement, an invalid output format, or an incorrect mapping yields `FAIL`; an ambiguous mapping yields `UNKNOWN`.

A.3 Selective Recovery and Cache Invalidation

Recovery Routing and Upstream Rechecks. Let J_t denote the verifier's proposed correction, containing a target, diagnosis, and revision. Given the stage outcomes $\boldsymbol{v}_t$, the controller produces

$$F_t = (a_t, \tau_t, \delta_t, \rho_t) = \mathrm{Route}(\boldsymbol{v}_t, J_t), \tag{10}$$

where a_t is the action, τ_t the target, δ_t the diagnosis, and ρ_t the proposed revision. ROUTE returns `ACCEPT` only when all three stages pass. Otherwise, it selects a recovery operator only when the diagnosis specifies a concrete revision; if not, it returns $\varnothing$.

A later-stage diagnosis may raise a specific concern about an earlier pass. Before recovery is selected, only the implicated upstream criterion is rechecked. A confirmed violation changes that criterion to `FAIL`, unresolved support to `UNKNOWN`, and a refuted concern leaves the cached `PASS` unchanged.

For a composite step, changing its internal operations, arguments, or dependencies requires REPLAN. REPERCEIVE repeats the existing composition without changing its acquisition definition.

Recovery Operators. `REPLAN` revises requirements or acquisition steps, including their operation, arguments, inputs, or dependencies. Evidence is retained only when its object identity, temporal and spatial references, scale, and acquisition conditions remain compatible with the revised plan. `REPERCEIVE` keeps the plan fixed and repeats the affected acquisition with the same operation, arguments, inputs, and dependencies; any change to that acquisition definition is handled by `REPLAN`. `REREASON` preserves both the plan and active evidence and regenerates the derivation and answer.

Dependency-Based Invalidation. Let Δ_t contain the records or steps directly changed or invalidated by recovery. The affected set is

$$\mathcal{I}_t = \Delta_t \cup \mathrm{Desc}(\Delta_t), \tag{11}$$

where Desc is the transitive closure under recorded input and support references. Superseded observations are removed from the active set but retained in the execution history. A cached criterion can be reused only if its supporting records are outside $\mathcal{I}_t$ and its justification remains valid for the revised state. Because every recovery generates a new reasoning trace and answer $(Z_{t+1}, \hat{y}_{t+1})$, checks on the previous reasoning or answer are never reused. The history H stores recovery feedback, whereas C stores criterion outcomes and their dependencies.

Budget Semantics. The budget T counts recovery actions rather than verification rounds. The initial state is verified at $t = 0$, and the state produced by the final permitted action is verified at $t = T$. A state that passes at $t = T$ is therefore returned as `VERIFIED`. If verification does not support another revision, or if the final state remains non-passing, ReTAC returns the current answer as `UNVERIFIED`. `VERIFIED` denotes acceptance under these criteria rather than a ground-truth label.

A.4 Complete Inference Procedure

Algorithm 1 combines native routing, residual refinement, ordered verification, and bounded recovery. VERIFY applies the stage-specific inputs above and reuses only valid cache entries; RESOLVECONCERNS performs targeted upstream rechecks when needed. RECOVER applies F_t and returns the revised plan, active evidence, and directly invalidated items Δ_t. During a recovery iteration, the Reasoner additionally receives the accumulated history H and the current feedback F_t; we denote this conditioning after a semicolon.

B Evaluation and Controlled Experimental Protocols

B.1 Shared Evaluation Protocol

We evaluate Qwen3-VL-8B, Gemini 3.1 Pro, and GPT-5.5 on VSI-Bench-Debiased and STI-Bench, using 32 uniformly sampled frames per video. The native call receives only the sampled video,

Algorithm 1 ReTAC inference with bounded selective recovery.

Require: $x = (V, q, \mathcal{O})$, model M, interface $\mathcal{T}$, recovery budget T
Ensure: answer $\hat{y}$ and status s

$S^{\text{nat}} \leftarrow \text{Native}_M(x)$
if $\mathcal{U}^{\text{nat}} = \varnothing$ **then**
  return $(\hat{y}^{\text{nat}}, \texttt{DIRECT})$
end if
$P_0 \leftarrow \text{Plan}_M(x, S^{\text{nat}}, \mathcal{T})$
$\mathcal{E}^{\text{new}} \leftarrow \text{Perceive}_M(x, P_0, \mathcal{T})$
$\mathcal{E}_0 \leftarrow \text{Merge}(\mathcal{E}^{\text{nat}}, \mathcal{E}^{\text{new}})$
$(Z_0, \hat{y}_0) \leftarrow \text{Reason}_M(q, \mathcal{O}, P_0, \mathcal{E}_0)$
$H \leftarrow (), C \leftarrow \varnothing$
for $t = 0, \ldots, T$ **do**
  $\Sigma_t \leftarrow (P_t, \mathcal{E}_t, Z_t, \hat{y}_t)$
  $(\boldsymbol{v}_t, J_t, C) \leftarrow \textsc{Verify}(x, \mathcal{T}, \Sigma_t, C)$
  $(\boldsymbol{v}_t, J_t, C) \leftarrow \textsc{ResolveConcerns}(x, \mathcal{T}, \Sigma_t, \boldsymbol{v}_t, J_t, C)$
  $F_t = (a_t, \tau_t, \delta_t, \rho_t) \leftarrow \text{Route}(\boldsymbol{v}_t, J_t)$
  if $a_t = \texttt{ACCEPT}$ **then**
    return $(\hat{y}_t, \texttt{VERIFIED})$
  end if
  if $a_t = \varnothing$ **or** $t = T$ **then**
    return $(\hat{y}_t, \texttt{UNVERIFIED})$
  end if
  $H \leftarrow \text{Append}(H, F_t)$
  $(P_{t+1}, \mathcal{E}_{t+1}, \Delta_t) \leftarrow \textsc{Recover}(x, M, \mathcal{T}, P_t, \mathcal{E}_t, H, F_t)$
  $\Delta_t \leftarrow \Delta_t \cup \{Z_t, \hat{y}_t\}$
  $\mathcal{I}_t \leftarrow \Delta_t \cup \text{Desc}(\Delta_t)$
  $C \leftarrow \text{Invalidate}(C, \mathcal{I}_t)$
  $(Z_{t+1}, \hat{y}_{t+1}) \leftarrow \text{Reason}_M(q, \mathcal{O}, P_{t+1}, \mathcal{E}_{t+1}; H, F_t)$
end for

question, and optional answer choices, without tool descriptions or outputs. All tool-enabled variants share the operation interface in Appendix C; the same base MLLM instantiates the Planner, Perceiver, Reasoner, and Verifier under role-specific instructions fixed across models (Appendix E).

The recovery budget T counts recovery actions rather than verification rounds: the initial state is verified at $t = 0$, and every recovered state is verified through the final permitted action at $t = T$. Unless stated otherwise, $T = 2$. All routing, component-replacement, answer-transition, and recovery-budget analyses use VSI-Bench-Debiased.

For the recovery-efficiency analyses on VSI-Bench-Debiased, we report the number of base-MLLM invocations averaged over the full 2,362-question evaluation split, including instances returned directly after native inference. For instance i, C_i counts the initial native call; all Planner and Reasoner calls; every Perceiver invocation, including multiple invocations within one plan step; all stage-aligned Verifier calls and any upstream rechecks triggered by the controller; and all failed calls and retries. Failed or retried invocations are therefore charged separately, and a direct-return instance contributes its native call. Deterministic spatial-tool executions that do not invoke the base MLLM are not counted as model calls.

For Tables 1, 2, and 3, each setting is evaluated in three runs on the same evaluation split under an identical configuration. We compute each task score and the benchmark overall score from the underlying per-question results separately for each run. Each run-level percentage is rounded to two decimal places before averaging across the three runs, and the resulting mean is again rounded to two decimal places for presentation. Overall scores are computed using the benchmark aggregation rule within each run and parenthesized differences are calculated from the displayed mean scores.

B.2 Benchmarks and Metrics

VSI-Bench-Debiased. We use the released VSI-Bench-Debiased v1 (Yang et al., 2025a; Brown et al., 2025) subset with 2,362 questions on indoor videos from ScanNet (Dai et al., 2017), ScanNet++ (Yeshwanth et al., 2023), and ARKitScenes (Baruch et al., 2021). The subset applies

question-specific manual filtering to reduce cases solvable from answer priors or prior knowledge without inspecting the video. Its eight tasks comprise four numerical tasks: object counting, absolute distance, object size, and room size; and four multiple-choice tasks: relative distance, relative direction, appearance order, and route planning.

Numerical tasks use mean relative accuracy (MRA). For each prediction, relative accuracy is evaluated at ten relative-error thresholds from 5% to 50% in 5% increments; MRA is the fraction of satisfied thresholds, averaged over questions. Multiple-choice tasks use answer accuracy. Relative-direction accuracy is first macro-averaged over the easy, medium, and hard subsets. The overall VSI-Bench-Debiased score is the unweighted mean of the eight task scores.

STI-Bench. We use the released STI-Bench test set, which contains 2,064 questions with five answer choices each, drawn from Omni6DPose (Zhang et al., 2024), ScanNet (Dai et al., 2017), and Waymo (Sun et al., 2020). It covers three static tasks (dimensional measurement, spatial relation, and 3D video grounding) and five dynamic tasks (displacement and path length, speed and acceleration, egocentric orientation, trajectory description, and pose estimation). Questions are generated from annotations in the source datasets and reviewed by human annotators to exclude inaccurate descriptions and cases with insufficient visual evidence. Answer choices reflect the measurement precision of each environment. We report accuracy for each task and overall accuracy across the full test set, equivalent to the mean task accuracy weighted by sample count.

C Spatial Tools and Operation Interface

The operation interface $\mathcal{T}$ exposes the spatial primitives used by residual planning and targeted perception. Each operation consumes sampled-video references or outputs from preceding steps and writes a traceable raw record that can be cited by an evidence record. Operations that combine 3D quantities require a common coordinate frame. Unless external calibration is available, reconstructed geometry has unknown metric scale and gravity; distances therefore remain in reconstruction units, and planar directions are interpreted as horizontal only when the supplied up direction is supported by evidence.

C.1 Visual Evidence Acquisition and 3D Localization

Frame Inspection (`inspect_frames`)

Input:
`frame_ids`: selected frame IDs from the sampled video.

Output:
`frame_ids` and `source=V`; the selected images are then provided to the Perceiver.

Description:
Retrieves selected frames for direct visual inspection. The operation supports localized checks of visibility, spatial relations, and temporal order without resampling the original video.

Object Detection (`object_detection`)

Input:
`objects`: at most eight requested object descriptions per call; `frame_ids`: selected frame IDs. The optional `box_threshold` and `text_threshold` default to 0.35 and 0.25, respectively, with both constrained to $[0, 1]$.

Output:
Each detection records `frame_id`, `object`, `box`, and `score`, where `box`$= [x_1, y_1, x_2, y_2]$ uses original-image pixel coordinates. The output also reports `earliest_detection_frame` over the queried frames.

Description:
Applies GroundingDINO to each requested object description on the selected frames. A missed detection does not establish absence, and detections in different frames do not by themselves establish instance identity.

Geometry Reconstruction (`reconstruct_geometry`)

Input:
`frame_ids`: at most 16 frame IDs.

Output:
`artifact`, `artifact_sha256`, `coordinate_frame`, `camera_extrinsics`, and `camera_intrinsics`. The saved NumPy archive stores `world_points`, `world_points_conf`, camera parameters, frame IDs, and original image sizes. The interface marks both `metric_scale` and `gravity` as `unknown`.

Description:
Applies VGGT to estimate dense 3D points and camera parameters. Preprocessing pads images without cropping their original content.

Object Localization (`localize_objects`)

Input:
`detection_step` and `geometry_step`: references to the detection and geometry steps, using their shared frames.

Output:
Each entry in `locations` contains the coordinate-wise median `center`, `extent_10_90` (the 90th–10th percentile span on each axis), and `valid_points`; `centers` provides the corresponding center list. All outputs retain the reconstruction coordinate frame.

Description:
Lifts 2D detections into the reconstruction. For each mapped box, it retains valid 3D points with confidence at least the within-box median and discards boxes with fewer than ten retained points. Because localization is box-based, the retained points can include background, and repeated detections of one object remain separate unless explicitly associated downstream.

C.2 Distance Operations

Euclidean Distance (`compute_distance`)

Input:
`a` and `b`: two 3D vector references that must share a coordinate frame.

Output:
`distance` and `units=input coordinate units`.

Description:
Resolves the two references and computes

$$d(a, b) = \|a - b\|_2.$$

Without metric calibration, the result remains in reconstruction units.

Closest-Point Distance (`compute_closest_point_distance`)

Input:
`object_a` and `object_b`: localized centers and extents in a shared coordinate frame; centers alone are also supported.

Output:
`distance`, `distance_type`, `closest_point_a`, `closest_point_b`, and `coordinate_frame`. The center-only fallback sets `distance_type=center_approximation`.

Description:
Estimates inter-object separation. When valid extents are available, it computes the minimum distance between the corresponding axis-aligned bounds. If only centers are available, it falls back to center distance. No metric conversion is applied to uncalibrated coordinates.

C.3 Coordinate Projections and Relative Direction

Planar Projection (`project_to_bev`)

Input:
`points`: referenced 3D points; `up`: a nonzero vector.

Output:
The projected 2D `points` and `basis`.

Description:
Projects the points onto the plane orthogonal to `up`. After normalizing `up`, the operation constructs an orthonormal basis for that plane. The projection represents a horizontal bird's-eye plane only when `up` is supported as the gravity direction; the operation neither estimates gravity nor renders an image.

Observer Pose (`observer_pose`)

Input:
`observer`, `facing`, `locations`, and `up`. The two localized references are supplied through `locations`: `observer` identifies the object beside the observer, and `facing` identifies the object toward which the observer is oriented; `up` supplies the up direction.

Output:
`observer_position`, `forward_vector`, `right_vector`, and `up_vector`.

Description:
Constructs an observer-centered basis from the two localized references. The forward direction points from the observer reference to the facing reference, and the right direction is derived from the forward and up vectors. It does not infer observer height, body pose, or gravity beyond the supplied evidence.

Observer-Frame Projection (`project_to_observer`)

Input:
`points`, `observer_position`, `forward_vector`, `right_vector`, and `up_vector`: 3D points, the observer position, and the observer basis vectors.

Output:
`observer_coordinates` and the basis vectors used for the transformation.

Description:
Expresses 3D points in the observer basis. For a point $\mathbf{p}$ and observer position $\mathbf{o}$, let $\mathbf{r} = \mathbf{p} - \mathbf{o}$. The returned coordinates are

$$x_{\text{right}} = \mathbf{r}^\top \mathbf{right}, \quad y_{\text{forward}} = \mathbf{r}^\top \mathbf{forward}, \quad z_{\text{up}} = \mathbf{r}^\top \mathbf{up}.$$

Relative-Direction Classification (`compute_rel_dir`)

Input:
`target`, `observer`, `forward`, `right`, `up`, and `back_angle`: the 3D target, observer reference, basis vectors, and back threshold. The default back threshold is 135°.

Output:
`direction`, `right_component`, `forward_component`, and `turn_angle_deg`.

Description:
Projects a 3D target into the observer frame. Let θ be the angle between the forward axis and the target direction. The target is classified as `back` when $\theta \geq$ `back_angle`; otherwise, the sign of the right component determines `left` or `right`.

Observer-Coordinate Direction Classification (`classify_dir`)

Input:
`observer`: coordinates already expressed in the observer frame; `back_angle`: the back threshold, with a default of 135°.

Output:
`left`, `right`, or `back`; `unknown` if no supported forward axis is available.

Description:
Applies the same classification rule as `compute_rel_dir` to coordinates already expressed in the observer frame, without assuming an image- or world-axis orientation when no supported forward axis is available.

C.4 Spatial Grouping and Cross-Frame Association

Spatial Clustering (`cluster_detections`)

Input:
`points`: 3D points; `radius`: a positive radius in the same coordinate units as the points.

Output:
`clusters`, `count`, and `radius`; cluster entries are zero-based indices into the input list.

Description:
Builds a graph over the points, connecting pairs whose Euclidean distance is at most `radius`, and returns its connected components as `clusters`. Because connectivity is transitive, a component can span more than `radius`, and its count should not be interpreted directly as an object count.

Cross-Frame Instance Association (`associate_instance`)

Input:
`locations`: localized detections in a shared coordinate frame; `radius`: the center-distance grouping radius.

Output:
Each entry in `instances` contains `instance_id`, `member_indices`, `frame_ids`, `cluster_center`, and `association_radius`.

Description:
Groups localized detections with the same object category whose centers fall within `radius`. The association is a geometric grouping cue rather than an identity guarantee; ambiguous category labels, coordinate alignment, or temporal continuity remain unresolved evidence conditions.

C.5 Composite Operations and Step Accounting

The operation interface provides registered primitive operations. The Planner may invoke a primitive directly or combine multiple primitives into a composite operation according to the requirements of the task. A composite operation is constructed during planning rather than selected from a fixed library of predefined compositions. For each internal operation, the Planner specifies its arguments, input references, dependencies, and documented output fields. Internal operations execute in dependency order and retain separate raw records for verification. The composite also declares the outputs available to subsequent plan steps. Each entry in the plan counts as one step, whether it invokes a single primitive or a composite operation. We encourage the Planner to use at most five steps per plan, but impose no fixed upper bound.

D Discussion and Limitations

D.1 Implications for Residual Test-Time Computation

Prior work shows that the effectiveness of test-time computation depends on problem difficulty, motivating adaptive computation strategies (Snell et al., 2025). Our analyses examine a related allocation question in video spatial reasoning: which unresolved requirements need additional computation, and which intermediate states need to be recomputed after a failure. On the fixed routing subsets, forcing supplementary acquisition for direct-return instances and suppressing refinement for refinement instances both reduce performance across the three base models (Fig. 3). Test-time scaling is allocation-sensitive: useful computation is determined by what remains unresolved for the current model–instance pair, rather than by a fixed inference depth. The recovery study shows the same pattern after refinement begins. Selective recovery reuses compatible plans and evidence, reducing repeated model calls relative to full restart, while larger recovery budgets do not yield monotonic gains (Fig. 5). These results support residual computation as an *allocation principle*: acquire only missing evidence and recompute only states invalidated by a localized failure.

D.2 Scope and Extensions

ReTAC's practical scope is set primarily by the quality of its state estimates and supplementary operations. A native call that incorrectly marks incomplete evidence as supported may terminate early, and using the same base MLLM for planning, perception, reasoning, and verification can preserve correlated errors across roles. Its supplementary evidence is likewise bounded by the operation interface; for example, the current VGGT-based geometry does not provide metric scale, consistent with the less uniform gains on metric tasks. More calibrated support estimates, independent verification signals, and uncertainty-aware perception outputs could strengthen routing and recovery without changing the residual formulation.

The empirical study covers three heterogeneous MLLMs and two video spatial reasoning benchmarks, with mean model calls as the primary compute proxy. We plan to extend the evaluation to other multimodal tasks to assess the broader applicability of residual allocation, and to measure token usage and tool latency for a more complete assessment of efficiency. We also plan to conduct targeted interventions to isolate the contributions of native state conditioning, evidence reuse, verification structure, and cached verification results under different computation budgets. Our current evaluation uses the original STI-Bench release. Given the coordinate and timestamp annotation issues reported by Sun et al. (2026), these results should be interpreted within the limitations of that release.

E Agent and Ablation Prompts

This appendix gives the role-specific prompts used by ReTAC and the two component-replacement ablations. Shared constraints are factored into prefixes: `NATIVE`, `PLANNER`, `PERCEIVER`, and `REASONER` prepend `COMMON`; `CHECK_COMMON` prepends `COMMON` to the shared verification instructions; each stage verifier then prepends `CHECK_COMMON`. All prompts are fixed across base models.

E.1 Shared Prompt Conventions

The shared prefix standardizes output format, identifiers, evidence bookkeeping, and answer representation across roles.

Prompt: `COMMON`

Return compact JSON matching the supplied schema, using actual findings rather than placeholders. Keep each explanation to one short sentence. Merge observations only when fact, object identity, time, and source agree; keep distinct or conflicting evidence separate.

Use unique IDs: r for requirements, e for evidence, s for steps, and z for reasoning claims. Frame IDs are ONE−BASED sampled−video positions; reference only supplied frames.

candidate_answer is a choice letter or one number encoded as a string in the requested units, or JSON null when unavailable. Never output the literal string ”null”. State uncertainty without inventing facts.

E.2 Native State Elicitation and Residual Routing

The native prompt produces the requirement–evidence state used to decide between direct return and residual refinement.

Prompt: `NATIVE`

Answer directly from the supplied video using native visual and spatial reasoning; no supplementary tools are available. Identify the concrete facts needed for the answer(e.g., identity, count, position, relation, dimension, or time) and record the corresponding native observations.

Label a requirement SUPPORTED only when the observations resolve it; otherwise label it UNRESOLVED and state the missing information. Requirements describe facts, not the final answer task. Every SUPPORTED requirement must reference matching evidence through requirement_id and evidence_ids. unresolved_requirements contains exactly the UNRESOLVED IDs. Preserve object identity, time, and spatial reference frame.

Native evidence uses method native, acquisition_step native_N0, and null tool−source fields. Put usable numerical quantities in claim.value and units, basis, and uncertainty in claim.text. Physical estimates based on familiar−object sizes are assumptions, not measurements. Return the best−supported candidate without duplicating the same observation across frames.

E.3 Residual Planning, Perception, and Reasoning

On the refinement branch, the three roles acquire only missing evidence, register it with provenance, and reason over the resulting active evidence set.

Residual Planner. The Planner converts unresolved requirements into an executable plan while preserving supported native state.

Prompt: `PLANNER`

Plan residual acquisition while preserving native requirements and observations. Retain all requirements and add any relevant requirement omitted by the native state. Every requirement targeted by supplementary acquisition, including additions during REPLAN, must appear in unresolved_requirements.

A plan step may invoke one registered primitive operation or a composition of registered primitives that you construct for the task. For each internal operation, declare its arguments, input references, dependencies, and documented output fields. Declare the outputs exposed by the composite and preserve a separate raw record for each internal operation.

Prefer plans with at most five steps. This is a recommendation, not a hard limit. If more than five steps are needed, review the plan for redundant acquisitions, unnecessary intermediate operations, or an unsuitable solution strategy. Revise the plan where appropriate and retain additional steps when they are necessary.

Use only registered primitives and documented output fields. Reference V, evidence IDs, or preceding operation outputs as inputs and declare dependencies. Measured arguments use {ref, field}.

object_detection takes category descriptions, not instance IDs. reconstruct_geometry uses at most 16 frames. localize_objects requires outputs from preceding object_detection and reconstruct_geometry operations, with both dependencies declared. These operations may appear in separate steps or within a composite step, provided that their dependency order is preserved. inspect_frames is not a detector.

Detection boxes are 4D pixel bounds; localization exposes 3D centers at locations.0.center. Reconstruction returns an artifact and camera matrices, not object positions, metric scale, or gravity. Never invent output paths. A missed detection does not establish absence. Use targeted frame inspection when sufficient.

On REPLAN, repair the diagnosed gap by changing the affected acquisition steps. Changing only expected_output or repeating an unchanged failed plan is not a repair.

Targeted Perceiver. The Perceiver turns one planned acquisition into source-linked evidence rather than a final task answer.

Prompt: `PERCEIVER`

Interpret only the current acquisition, its input records, and source frames. Produce observations for the target requirements, not the final answer. Cite raw tool−record IDs, not step IDs. For a tool−extracted value, set tool_field to the actual output−relative path and claim.value to that value. Direct visual claims use tool_field null and may use tool_record_id null. Give numerical quantities explicit units and a short basis in claim.text.

Preserve object identity, time, coordinate frame, and scale; report unresolved requirements and conflicts.

A failed tool is not evidence. Detection alone establishes neither unique identity nor absence; first−appearance claims require checking preceding sampled frames. VGGT has unknown metric scale and gravity, so metric conversion or a horizontal reference requires supporting evidence. Approximate physical estimates must state the visible cue and assumed reference size.

Evidence-Grounded Reasoner. The Reasoner derives the candidate from active evidence without introducing new visual observations.

Prompt: `REASONER`

Reason only from active evidence. Each claim's support_ids must reference active evidence or earlier supported claims. Track identity, time, coordinates, units, and conflicts. Return the best−supported candidate and retain unresolved requirements or conflicts.

For arithmetic, provide an operator, traced operands, and result; semantic claims use calculation null. Evidence operands reference claim.value, computed operands reference calculation.result, and every operand appears in support_ids. Do not create a numeric claim.value for a semantic conclusion solely to make it referenceable; cite the underlying numerical evidence instead.

final_conclusion and candidate_answer must agree on the option or quantity. Put uncertainty in the reasoning, not an answer range. Retain estimates supported by explicit evidence and assumptions when exact calibration is unavailable.

E.4 Stage-Aligned Verification

Verification mirrors the planning–perception–reasoning decomposition; each stage receives only the inputs needed for its criteria in Appendix A.2.

Shared Verification Instructions. The shared verifier prefix defines verdict semantics, recovery actions, and targeted upstream rechecks.

Prompt: `CHECK_COMMON`

Evaluate every named criterion independently: PASS requires support, FAIL a definite error, and UNKNOWN insufficient support to decide. Identify specific targets. For FAIL or UNKNOWN,

propose a concrete supported correction when available; otherwise set action to null and revision to empty.

REPERCEIVE repeats the existing operation, arguments, inputs, dependencies, and frame selection. Changed acquisition requires REPLAN; REREASON changes reasoning only. Recovery target lists contain IDs.

If a later check raises a specific concern about an earlier PASS, report upstream_concern with stage, criterion, target, reason, and revision. The controller rechecks that criterion before routing recovery; do not overturn a PASS on a hypothesis.

Planning Verifier. The planning verifier checks plan completeness and feasibility without seeing observations, reasoning, or the candidate answer.

Prompt: `PLAN_CHECK`

You receive the question, options, plan, tool interfaces, and reference metadata, but NO video, observations, reasoning, or answer.

plan_completeness: Check whether the plan covers the entities, relations, and references needed for the answer; could an omitted fact change the answer while planned facts remain fixed?

operation_feasibility: Check primitive operations, arguments, input references, output capability, and dependency order both within composite steps and across plan steps. Native−supported requirements need no reacquisition. inspect_frames can acquire spatial or temporal observations, not only object presence. Judge what execution COULD supply; withheld observations are not a planning failure. Metric quantities require a feasible calibration or estimation procedure.

A supported planning failure proposes REPLAN; without a concrete supported revision, stop unverified.

Perception Verifier. The perception verifier validates every active evidence record against its cited source before checking requirement coverage.

Prompt: `PERCEPTION_CHECK`

Compare all active native and supplementary evidence with the cited frames and raw operation records. Check source_validity and requirement_coverage separately. Preserve object identity, time, coordinate frame, and scale.

source_validity: Check every active evidence record. Validate its frame references, operation records, and cited output fields, then determine whether the source supports the observation. Return PASS when support is established, FAIL for a definite inconsistency, and UNKNOWN when the source is inaccessible or support is ambiguous. Include an individual check for every active evidence ID in the observations field defined by the supplied schema.

requirement_coverage: Check whether each requirement is resolved by valid active evidence. Return PASS for a resolved requirement. Return FAIL when the required evidence is missing or contradicted. Return UNKNOWN when support or relevance remains uncertain. A record whose source validity is UNKNOWN does not by itself establish either coverage or a definite coverage failure. Consider any other active evidence for the same requirement.

Propose REPERCEIVE when repeating the affected acquisition with unchanged operations, arguments, inputs, dependencies, and frame selection can repair the evidence. If the acquisition must change, or a native error has no corresponding acquisition step, report a targeted planning concern with a concrete proposed revision. The controller rechecks the implicated planning criterion before selecting REPLAN. If no concrete supported revision is available, set action to null and revision to empty.

Reasoning Verifier. The reasoning verifier checks the evidence-to-answer derivation after planning and perception pass.

Prompt: `REASON_CHECK`

Check evidence_support, inference_validity, and answer_consistency along evidence −> cited claims −> conclusion −> answer. Validate premises, semantic or geometric inferences, and calculations. Map conclusions to option IDs and check units and requested rounding.

A plausible answer does not excuse unsupported premises. A specific evidence gap first raises a perception concern; if the plan and evidence are valid, reasoning, calculation, mapping, or formatting errors propose REREASON.

E.5 Component-Replacement Ablation Prompts

These prompts implement the component replacements in Sections 4.3 and 4.4: `GENERAL_PLANNING` removes native-state conditioning, while `GENERAL_REFLECTION` replaces stage-aligned verification and localized recovery with a trajectory-level decision.

General Planning. The general planner decides whether to use supplementary computation without access to native support labels, native evidence, or native unresolved requirements.

Prompt: `GENERAL_PLANNING`

You are a general adaptive planner for video spatial reasoning. You receive the video, question, answer options, operation interfaces, and descriptions of the facts needed to answer. Decide whether supplementary computation is useful from these inputs.

Return RETURN_NATIVE when direct reasoning from the input appears sufficient. The controller then returns the stored native candidate unchanged; do not generate a new answer. You are not given that candidate, native support labels, native observations, or native unresolved requirements.

Otherwise return SUPPLEMENT with a complete plan covering all facts needed for the answer. Retain supplied requirement IDs and add missing requirements. Assume no requirement is already supported: every downstream fact must be established by the new acquisition because native observations are unavailable.

Use ordered steps, including prerequisites, with registered operations and documented arguments and outputs. Each step specifies target_requirement, operation, arguments, input_references, dependencies, and expected_output. Inputs may reference V or earlier steps, not native evidence. Use {ref, field} for prior outputs, declare dependencies, and never invent output fields. Targeted frame inspection is available. Detection alone establishes neither identity nor absence; reconstructed geometry has no metric scale or gravity.

Return compact JSON with exactly these top−level fields:
− decision: RETURN_NATIVE or SUPPLEMENT.
− rationale: one short sentence explaining the computation decision.
− plan: null for RETURN_NATIVE; otherwise an object with requirements and steps.

For SUPPLEMENT, address the full task rather than a presumed subset of native failures. Describe acquisition, not anticipated observations or an answer. Reference only supplied sampled−frame IDs.

General Reflection. The general reflector makes one accept-or-retry decision over the full trajectory instead of localizing a failure to a specific stage.

Prompt: `GENERAL_REFLECTION`

You are a general reflector for video spatial reasoning. Review the current candidate and the trajectory as a whole using the question, options, fixed native output, current plan, active evidence, reasoning,

supplied video sources, raw tool records, and previous feedback. Decide whether the current answer is sufficiently supported.

Return ACCEPT when the answer is adequately supported. Otherwise return RETRY with brief, concrete feedback on what the next attempt should address. Do not assume unseen source content or treat previous feedback as new evidence.

Use one overall decision. Do not output per−stage verdicts, criterion scores, structured correction targets, or stage−specific action codes. Do not rewrite the candidate. ACCEPT returns it unchanged. RETRY reruns supplementary planning, perception, and reasoning from the fixed native attempt using accumulated feedback; the controller enforces the retry budget.

Return only compact JSON with exactly two fields:
– decision: the literal string ACCEPT or RETRY.
– feedback: a brief explanation, at most two sentences.

F WORKED EXAMPLE: PERCEPTION ERROR AND SELECTIVE RECOVERY

We trace a refinement trajectory in which the residual Perceiver produces a source-inconsistent temporal observation. The example follows the state definitions, operation interface, and prompts in Appendices A–E. Unlike Case 2 in Fig. 6, which corrects a reasoning error with `REREASON`, this trajectory fails at perception and triggers `REPERCEIVE`. YAML records are abbreviated, with model outputs, raw operation records, and controller updates separated.

F.1 EXAMPLE INPUT AND CONVENTIONS

We use an appearance-order query with directly traceable temporal evidence. Let $x = (V, q, \mathcal{O})$, where V contains 32 ordered sampled frames, $q =$ "Which object appears first in the provided frame sequence: a table, a lamp, or a stool?", and $\mathcal{O} = \{\mathrm{A} : \text{table}, \mathrm{B} : \text{lamp}, \mathrm{C} : \text{stool}\}$. The corresponding instances are `table_1`, `lamp_1`, and `stool_1`, whose first appearances in this trace are sampled frames 8, 24, and 4. Requirements `r_table`, `r_lamp`, and `r_stool` encode these facts. Frame indices are one-based, ranges are inclusive, and temporal claims refer only to the sampled sequence. The recovery budget is $T = 2$.

F.2 NATIVE STATE AND INITIAL RESIDUAL REFINEMENT

The native state resolves the table and lamp appearances but leaves the stool appearance unresolved. Residual refinement therefore targets only `r_stool`.

1. Native state and routing

Input.
The native call receives only x and uses base model M, without tool descriptions or tool outputs. Its producing-step identifier is `native_N0`.

Model output.
The call returns $S^{\mathrm{nat}} = (\mathcal{R}^{\mathrm{nat}}, \mathcal{E}^{\mathrm{nat}}, \mathcal{U}^{\mathrm{nat}}, \hat{y}^{\mathrm{nat}})$:

```
requirements:
  - id: r_table
    object_id: table_1
    status: SUPPORTED
    evidence_ids: [e_table]
  - id: r_lamp
    object_id: lamp_1
    status: SUPPORTED
    evidence_ids: [e_lamp]
  - id: r_stool
    object_id: stool_1
```

```
    status: UNRESOLVED
    evidence_ids: []
    reason: Cannot establish the first appearance.
evidence:
  - id: e_table
    requirement_id: r_table
    object_id: table_1
    claim:
      value: 8
      text: table_1 first appears at sampled frame 8.
    sources:
      video: V
      frame_range: [1, 8]
      tool_record_id: null
      tool_field: null
    method: native
    acquisition_step: native_N0
    input_record_ids: []
  - id: e_lamp
    requirement_id: r_lamp
    object_id: lamp_1
    claim:
      value: 24
      text: lamp_1 first appears at sampled frame 24.
    sources:
      video: V
      frame_range: [1, 24]
      tool_record_id: null
      tool_field: null
    method: native
    acquisition_step: native_N0
    input_record_ids: []
unresolved_requirements: [r_stool]
candidate_answer: null
```

Controller update.
The labels give $\mathcal{U}^{\text{nat}} = \{\texttt{r_stool}\}$, so the instance enters refinement. `SUPPORTED` is a routing label, not a verification result; the native observations remain provisional. Because no native answer is established, $\hat{y}^{\text{nat}} = \varnothing$ is encoded as `null`. An empty $\mathcal{U}^{\text{nat}}$ would instead return the native candidate with status `DIRECT`.

2. Residual plan

Input.
The Planner receives $(x, S^{\text{nat}}, \mathcal{T})$. Here `object_detection(objects, frame_ids)` returns frame-indexed detections and `earliest_detection_frame`; preceding sampled frames must still be inspected because a missed detection does not establish absence.

Model output.
The Planner returns $P_0 = (\mathcal{S}_0, \mathcal{R}_0, \mathcal{U}_0)$:

```
requirements: [r_table, r_lamp, r_stool]
unresolved_requirements: [r_stool]
steps:
  - id: s_stool
    target_requirement: [r_stool]
    operation: object_detection
    arguments:
      objects: [stool]
      frame_ids: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
    input_references: [V]
```

```
    dependencies: []
    expected_output: >
      Frame-indexed stool detections within sampled frames 1--12.
      Use the detections and source frames to establish the first
      appearance if supported; otherwise keep r_stool unresolved.
```

Controller update.
No requirement is added, so $\mathcal{R}_0 = \mathcal{R}^{\mathrm{nat}}$ and $\mathcal{U}_0 = \mathcal{U}^{\mathrm{nat}}$. Only `r_stool` is reacquired. Source support is decided after execution; changing the queried frames would alter the plan and require `REPLAN`.

3. Supplementary perception and evidence merge

Input.
The Perceiver executes `s_stool`, retains the raw record, and interprets it with the queried source frames.

Raw operation record.

```
tool_record:
  id: D0
  acquisition_step: s_stool
  video: V
  frame_ids: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
  earliest_detection_frame: 4
```

Perceiver output.
The Perceiver incorrectly records frame 10 despite citing a raw field whose value is 4:

```
new_evidence:
  - id: e_stool_v0
    requirement_id: r_stool
    object_id: stool_1
    claim:
      value: 10
      text: stool_1 first appears at sampled frame 10.
    sources:
      tool_record_id: D0
      tool_field: earliest_detection_frame
      video: V
      frame_range: [1, 12]
    method: object_detection_with_source_inspection
    acquisition_step: s_stool
    input_record_ids: [D0]
acquisition_failures: []
reported_unresolved_requirements: []
reported_conflicts: []
```

Controller update.
The active evidence becomes

$$\mathcal{E}_0 = \{\texttt{e_table}, \texttt{e_lamp}, \texttt{e_stool_v0}\}.$$

The raw record `D0` remains in the execution history. Empty unresolved/conflict lists do not validate the observation; verification checks source validity and requirement coverage separately.

4. Initial reasoning

Input.
The Reasoner receives $(q, \mathcal{O}, P_0, \mathcal{E}_0)$.

Model output.
It returns $(Z_0, \hat{y}_0)$:

```
reasoning:
  - id: z_table_before_stool_v0
    support_ids: [e_table, e_stool_v0]
    calculation:
      operator: "<"
      operands:
        - {ref: e_table, field: claim.value}
        - {ref: e_stool_v0, field: claim.value}
      result: true
    conclusion: The table appears before the stool.
  - id: z_table_before_lamp_v0
    support_ids: [e_table, e_lamp]
    calculation:
      operator: "<"
      operands:
        - {ref: e_table, field: claim.value}
        - {ref: e_lamp, field: claim.value}
      result: true
    conclusion: The table appears before the lamp.
  - id: z_answer_v0
    support_ids: [z_table_before_stool_v0,
                  z_table_before_lamp_v0]
    calculation: null
    conclusion: "A: table"
candidate_answer: A
unresolved_requirements: []
unresolved_conflicts: []
```

Controller update.
Set $t = 0$, $H = ()$, and $C = \varnothing$. The comparisons are consistent with the active values, but the stool premise is source-inconsistent; the candidate remains provisional.

F.3 Stage-Aligned Verification and Error Localization

Verification follows planning–perception–reasoning order. The plan passes, but the stool observation contradicts its cited source, so verification stops at perception.

5a. Planning verification

Check input.
The planning verifier receives only $(q, \mathcal{O}, P_0, \mathcal{T})$, without the video, observation contents, reasoning trace, or candidate answer. Deterministic checks use the declared operation arguments, references, and dependencies.

Check output.

```
planning:
  plan_completeness:
    result: PASS
    reason: >
      The three requirements cover the first appearance of each
      listed object under the same temporal reference.
  operation_feasibility:
    result: PASS
    rule_checks:
      registered_operation: PASS
      arguments: PASS
```

```
      input_references_and_dependencies: PASS
    reason: >
      object_detection can provide stool detections over the
      declared sampled frames; source support is checked later.
  result: PASS
  origin: fresh
```

Controller update.
Record $v_0^{\text{plan}} = \texttt{PASS}$. Feasibility does not validate `e_stool_v0`.

5b. Perception failure and recovery routing

Check input.
The perception verifier receives $P_0, \mathcal{E}_0$, the cited sampled frames, raw record `D0`, and the relevant interface definition. Native and supplementary observations are checked under the same source-validity criterion.

Check output.

```
perception:
  source_validity:
    result: FAIL
    observations:
      e_table:
        result: PASS
        reason: Frame 8 shows table_1; frames 1--7 do not.
      e_lamp:
        result: PASS
        reason: Frame 24 shows lamp_1; frames 1--23 do not.
      e_stool_v0:
        result: FAIL
        rule_reason: >
          claim.value is 10, but D0.earliest_detection_frame is 4.
        source_reason: >
          Frame 4 shows stool_1 and contradicts a first
          appearance at frame 10.
  requirement_coverage:
    result: FAIL
    requirement_results:
      r_table: PASS
      r_lamp: PASS
      r_stool: FAIL
    reason: r_stool has no source-valid active observation.
  result: FAIL
  origin: fresh
reasoning:
  result: null
  origin: not_run
```

Thus $\boldsymbol{v}_0 = (\texttt{PASS}, \texttt{FAIL}, \text{not run})$. Reasoning verification is not executed after the perception failure; `not_run` is distinct from `UNKNOWN`.

Diagnostic proposal J_0**.**

```
target: [e_stool_v0]
diagnosis: >
  The observation misreports the cited earliest detection and
  ignores the source-supported occurrence at sampled frame 4.
revision: >
  Repeat s_stool with the same acquisition definition, inspect
  the cited detection and preceding sampled frames, replace the
  invalid stool observation, and regenerate dependent reasoning.
```

Controller output.
$\mathrm{Route}(\boldsymbol{v}_0, J_0)$ returns $F_0 = (a_0, \tau_0, \delta_0, \rho_0)$:

```
decision: REPERCEIVE
target: [e_stool_v0]
diagnosis: >
  e_stool_v0 is inconsistent with its cited raw field and frame.
revision: >
  Repeat s_stool with unchanged operation, arguments, inputs,
  and dependencies; correct the stool observation and regenerate
  the dependent reasoning and answer.
```

Controller update.
The same acquisition can repair the error, so no planning recheck is needed. Since $t = 0 < T$, F_0 is appended to H. Planning checks and source-validity checks for `e_table` and `e_lamp` remain cacheable; checks depending on `e_stool_v0` are invalidated.

F.4 Selective Recovery, Reverification, and Final Return

Selective recovery replaces the invalid stool observation and recomputes its dependents while reusing the plan and unaffected native evidence.

6. Reperception and dependency update

Recovery input.
The recovery call receives $(x, M, \mathcal{T}, P_0, \mathcal{E}_0, H, F_0)$ and keeps $P_1 = P_0$. Step `s_stool` therefore uses the same operation, frame IDs, inputs, and dependencies.

Raw operation record.
Repeating the acquisition produces `D1`:

```
tool_record:
  id: D1
  acquisition_step: s_stool
  video: V
  frame_ids: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
  earliest_detection_frame: 4
```

The raw output is unchanged; the correction comes from the Perceiver's revised interpretation and source-frame inspection.

Corrected Perceiver output.

```
new_evidence:
  - id: e_stool_v1
    requirement_id: r_stool
    object_id: stool_1
    claim:
      value: 4
      text: stool_1 first appears at sampled frame 4.
    sources:
      tool_record_id: D1
      tool_field: earliest_detection_frame
      video: V
      frame_range: [1, 4]
    method: object_detection_with_source_inspection
    acquisition_step: s_stool
    input_record_ids: [D1]
acquisition_failures: []
reported_unresolved_requirements: []
reported_conflicts: []
```

Controller update.
The revised active evidence is

$$\mathcal{E}_1 = \{\texttt{e_table}, \texttt{e_lamp}, \texttt{e_stool_v1}\}.$$

The source range $[1, 4]$ supports the corrected claim, while the planned acquisition remains frames 1–12. `e_stool_v0` leaves the active set but remains in the history with `D0`; the two native observations remain active.

Reasoner output.
Regenerating reasoning from $(q, \mathcal{O}, P_1, \mathcal{E}_1; H, F_0)$ gives $(Z_1, \hat{y}_1)$:

```
reasoning:
  - id: z_stool_before_table_v1
    support_ids: [e_stool_v1, e_table]
    calculation:
      operator: "<"
      operands:
        - {ref: e_stool_v1, field: claim.value}
        - {ref: e_table, field: claim.value}
      result: true
    conclusion: The stool appears before the table.
  - id: z_stool_before_lamp_v1
    support_ids: [e_stool_v1, e_lamp]
    calculation:
      operator: "<"
      operands:
        - {ref: e_stool_v1, field: claim.value}
        - {ref: e_lamp, field: claim.value}
      result: true
    conclusion: The stool appears before the lamp.
  - id: z_answer_v1
    support_ids: [z_stool_before_table_v1,
                  z_stool_before_lamp_v1]
    calculation: null
    conclusion: "C: stool"
candidate_answer: C
unresolved_requirements: []
unresolved_conflicts: []
```

Cache update.
Only checks whose dependencies remain unchanged are retained:

```
retained_checks:
  - planning.plan_completeness
  - planning.operation_feasibility
  - source_validity(e_table)
  - source_validity(e_lamp)
invalidated_checks:
  - source_validity(e_stool_v0)
  - perception.requirement_coverage
requires_verification:
  - source_validity(e_stool_v1)
  - perception.requirement_coverage
  - reasoning.evidence_support
  - reasoning.inference_validity
  - reasoning.answer_consistency
```

No reasoning check is reusable because the initial verification stopped at perception. Set $t = 1$ and verify the revised state.

7. Reverification and return

Input.
The verifier receives $\Sigma_1 = (P_1, \mathcal{E}_1, Z_1, \hat{y}_1)$ with the still-valid cache entries. The unchanged planning criteria remain `PASS`; perception first verifies the replacement observation and coverage before reasoning is checked.

Check output.

```
planning:
  plan_completeness: {result: PASS, origin: cached}
  operation_feasibility: {result: PASS, origin: cached}
  result: PASS
perception:
  source_validity:
    result: PASS
    observations:
      e_table: {result: PASS, origin: cached}
      e_lamp: {result: PASS, origin: cached}
      e_stool_v1:
        result: PASS
        origin: fresh
        reason: >
          claim.value=4 matches D1.earliest_detection_frame;
          frame 4 shows stool_1 and frames 1--3 do not.
  requirement_coverage:
    result: PASS
    origin: fresh
    requirement_results:
      r_table: PASS
      r_lamp: PASS
      r_stool: PASS
  result: PASS
reasoning:
  evidence_support:
    result: PASS
    reason: >
      Every critical claim cites active evidence or an earlier
      supported claim, and each premise is source-supported.
  inference_validity:
    result: PASS
    reason: >
      Both 4 < 8 and 4 < 24 hold for the cited evidence values.
  answer_consistency:
    result: PASS
    reason: The final conclusion and candidate both map to C.
  result: PASS
  origin: fresh
```

Controller output and return.
Now $\boldsymbol{v}_1 = (\texttt{PASS}, \texttt{PASS}, \texttt{PASS})$:

```
decision: ACCEPT
target: null
diagnosis: All required checks pass.
revision: null
answer: C
status: VERIFIED
correction_rounds: 1
```

The corrected state is accepted after one recovery action. The unused budget is not consumed; a state that passes at $t = T$ is still accepted because T limits recovery actions, not the final verification.

This trajectory isolates the recovery scope: the perception error invalidates the stool observation, requirement coverage, and dependent reasoning, while the plan and unrelated native evidence remain reusable.