SPATIALCREW: MULTI-AGENT FRAMEWORK FOR RELIABLE TOOL-AUGMENTED SPATIAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Tool augmentation can improve the spatial reasoning of vision-language models (VLMs), but erroneous intermediate outputs can propagate through otherwise valid reasoning chains. Our empirical analysis identifies this error cascade as a major source of failure and shows that verification becomes less effective as reasoning spans more tools and domains. We propose SPATIALCREW, a multi-agent framework that combines domain-specific Write–Execute–Verify–Adjust (WEVA) loops with coordinated reasoning across perception, geometry, and temporal specialists. This design keeps local verification focused while checking consistency between domains. On 20 benchmarks with Gemma4-31B, SPATIALCREW reduces the error cascade rate (ECR) from the five tool-augmented baselines' mean of 78.0% to 23.4%, a 54.6-percentage-point reduction. Its overall benchmark macro-average is 62.6, exceeding the mean of all six baselines by 9.6 points and the strongest baseline by 2.7 points. Across all five evaluated backbones, SPATIALCREW achieves a mean score of 62.9, an average gain of 9.1 points over the corresponding no-tool models, and a mean ECR of 24.7%. These results highlight intermediate-output quality control as a practical route to reliable tool-augmented spatial reasoning. Code: `https://anonymous.4open.science/r/SpatialCrew-B650`.

1 INTRODUCTION

Spatial reasoning is central to visual intelligence, yet recent vision-language models (VLMs) (Bai et al., 2025; Abd et al., 2026) struggle on spatial benchmarks (Fu et al., 2024), as depth, 3D geometry, and motion are difficult to learn from 2D image-text pretraining alone. Data-driven approaches fine-tune VLMs on spatial annotations (Chen et al., 2024; Cheng et al., 2024; Song et al., 2025), but face high annotation costs and limited generalization beyond the training distribution. Tool-augmented approaches instead obtain spatial information at inference time through depth estimators (Yang et al., 2024), segmentation models (Ravi et al., 2025), and 3D reconstruction, offering greater flexibility.

Existing tool-augmented methods differ mainly in their execution paradigm (Figure 1). **One-shot execution** methods (Gupta & Kembhavi, 2023; Surís et al., 2023; Luo et al., 2026; Marsili et al., 2025) generate complete programs from tool descriptions and execute them without inspecting intermediate outputs during execution. **Step-by-step execution** methods invoke tools incrementally through two types of interfaces. Structured-API methods (Chen et al., 2025a;b) select predefined functions and arguments within fixed input/output schemas, limiting custom logic. Code-based methods (Cho et al., 2026) execute arbitrary Python in a persistent environment, enabling flexible tool calls.These methods share the goal of improving tool orchestration and provide effective mechanisms that help agents select appropriate tools, order calls, and combine their outputs into a final answer.

However, all these methods overlook a more fundamental issue: **the correctness of intermediate tool outputs**. Spatial tools such as depth estimators, segmentation models, and 3D reconstruction pipelines inherently produce noisy and uncertain outputs. Once an intermediate result is erroneous and goes unchecked, all downstream computations are built on a faulty foundation, and the error propagates along the reasoning chain until it corrupts the final answer. We refer to this phenomenon as **Error Cascade** (Figure 3). Investigating this problem is essential, because such failures can occur even when tool selection and invocation order are correct, motivating a systematic analysis of intermediate-output reliability. To understand the status of error cascade in existing methods, we evaluate representative methods using error cascade rate (ECR) and correction rate (CR), revealing

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.

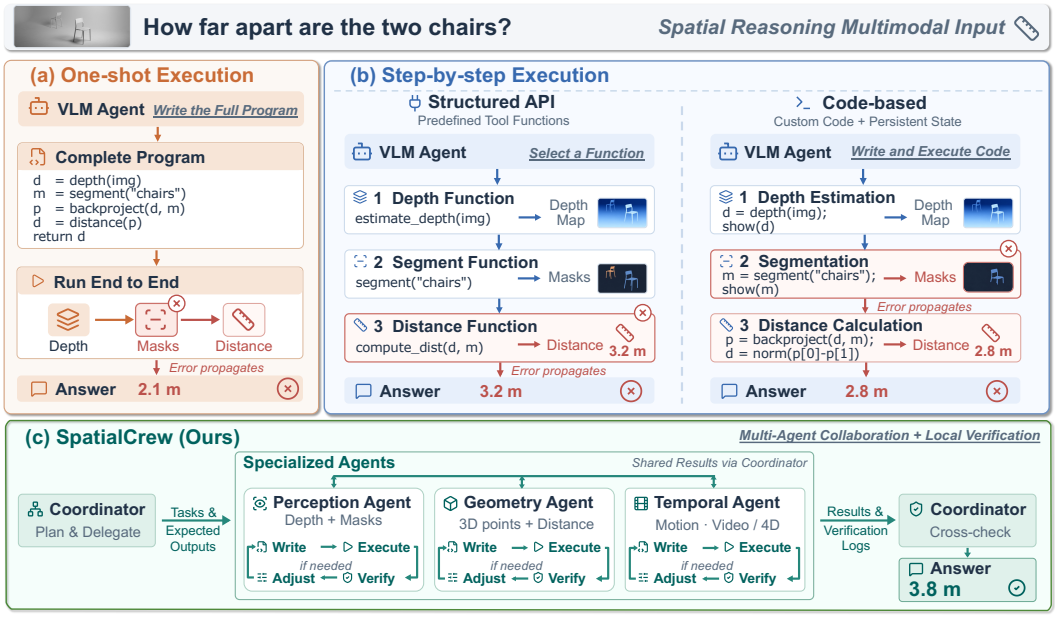

Figure 1: Comparison of spatial reasoning paradigms. **(a)** One-shot execution; **(b)** step-by-step execution via structured APIs or code; and **(c) SPATIALCREW**, where a coordinator delegates tasks to perception, geometry, and temporal agents, each using a Write–Execute–Verify–Adjust (WEVA) loop for verification and refinement. Red arrows illustrate errors and their propagation.

three findings (Figures 2 & 3): ❶ error cascades dominate incorrect answers, accounting for 79.8% of failures on average across four methods, compared with 8.8% for tool-orchestration errors; ❷ verification reduces ECR and improves CR, with step-by-step execution benefiting more than one-shot execution; ❸ correction ability declines sharply as tool count and reasoning complexity grow, with context overload leading to verification omission and tool misuse. These findings identify intermediate-output quality control as a key reliability bottleneck: verification is effective, but its benefits diminish with complexity. Suppressing error cascades therefore requires flexible, closed-loop verification and a manageable verification scope as tool count and domain complexity grow.

Guided by these findings, we propose **SPATIALCREW**, a multi-agent framework for tool-augmented spatial reasoning that addresses error cascades through two complementary mechanisms (Figure 1(c)). First, each specialist follows a **Write–Execute–Verify–Adjust (WEVA)** cycle, checking code outputs against domain-specific rules and revising and re-executing failed steps before passing results downstream. Second, decomposing reasoning by domain limits each specialist's tools and verification responsibilities, keeping its verification scope manageable. The framework comprises four agents: ❶ a Coordinator Agent that decomposes questions into subtasks with expected output specifications, dispatches work, checks cross-agent consistency, and synthesizes answers; ❷ a Perception Agent for depth estimation and segmentation; ❸ a Geometry Agent for 3D reconstruction, coordinate transformations, and geometric computations; ❹ a Temporal Agent for motion tracking and cross-frame correspondence. We evaluate SPATIALCREW on 20 benchmarks spanning single-image, multi-view, and video settings with five VLM backbones. With Gemma4-31B, it reduces ECR from the five tool-augmented baselines' mean of 78.0% to 23.4% (54.6 pp) and achieves an overall macro-average of 62.6, exceeding the six-baseline mean of 53.0 by 9.6 points. Across all five backbones, SPATIALCREW achieves a mean score of 62.9 and a mean ECR of 24.7%, outperforming all existing baselines. Our work enables agents to orchestrate tools efficiently and reliably for spatial reasoning.

Our contributions are threefold: ❶ we identify the overlooked problem of error cascade in tool-augmented spatial reasoning and show how verification effectiveness degrades with reasoning complexity (Section 3); ❷ we introduce SPATIALCREW, combining domain-specific WEVA loops with coordinated multi-agent reasoning to keep verification focused (Section 4); and ❸ we conduct comprehensive evaluation on diverse spatial reasoning benchmarks with multiple VLM backbones. SPATIALCREW reduces the error cascade rate by over 54.6-pp and outperforms all existing baselines, with the largest gains on tasks requiring cross-domain reasoning chains (Sections 5 and 6).

2 Background and Related Work

2.1 Data-Driven Spatial Reasoning

Most VLMs are pretrained on 2D image-text pairs and lack grounded 3D spatial understanding. To bridge this gap, SpatialVLM (Chen et al., 2024) curates two billion spatial QA pairs, SpatialRGPT (Cheng et al., 2024) trains on depth-annotated data for region-level reasoning, and RoboSpatial (Song et al., 2025) collects dense 3D annotations for robotic manipulation. These methods are effective within their training distributions but suffer from expensive annotation costs and limited generalization to new scenarios and tasks.

2.2 Tool-Augmented Spatial Reasoning

Tool augmentation obtains spatial information from external tools at inference time. Spatial reasoning methods follow two execution paradigms. **One-shot execution** methods generate complete programs before invoking tools: Visual Programming (Gupta & Kembhavi, 2023) and ViperGPT (Surís et al., 2023) compose tool calls into Python scripts, pySpatial (Luo et al., 2026) adds spatial tool libraries, and VADAR (Marsili et al., 2025) incorporates visual annotations. Agents cannot inspect or react to intermediate outputs during execution. **Step-by-step execution** methods invoke tools incrementally. Structured-API approaches, including SpaceTools (Chen et al., 2025a) and GCA (Chen et al., 2025b), expose fixed-signature functions that restrict agents to predefined computations; an API exposing only average distance, for example, cannot directly compute nearest-point distance. Code-based approaches such as SpatialClaw (Cho et al., 2026) instead provide a persistent Python kernel for arbitrary code execution, flexible tool composition, post-processing, and verification logic. These methods provide effective mechanisms for tool orchestration, while the correctness of intermediate outputs remains insufficiently examined. We investigate this issue empirically in the next section.

3 Empirical Analysis on Error Cascade

Tool-augmented spatial reasoning relies on noisy and uncertain tool outputs. Unchecked intermediate errors can propagate through downstream computations and corrupt the final answer, a phenomenon we call **Error Cascade**. Existing methods support effective tool orchestration, but intermediate-output reliability remains underexplored. Since such failures can occur even when tools are correctly selected and invoked, we conduct an empirical analysis of error cascades across representative methods.

Setup. We select representative methods from each execution paradigm: Visual Programming (Gupta & Kembhavi, 2023) and pySpatial (Luo et al., 2026) (one-shot execution), SpaceTools (Chen et al., 2025a) and SpatialClaw (Cho et al., 2026) (step-by-step execution with structured APIs and code-based execution, respectively). All methods use the same VLM backbone (Gemma4-31B (Abd et al., 2026)) and tool set. We select three benchmarks to cover reasoning chains of varying complexity: SPBench (Li et al., 2025b) (single-image), SPAR-Bench (Zhang et al., 2025a) (multi-view), and VSI-Bench (Yang et al., 2025a) (video), where single-image tasks typically involve 2–3 tool invocations, multi-view tasks 4–5, and video tasks 6 or more. Since the analysis requires detailed manual annotation of reasoning traces, we randomly sample 100 questions from each benchmark, yielding 300 questions in total (1,200 traces). The verification-augmented comparison uses the same questions in additional runs; these runs are not included in the 1,200-trace count.

Evaluation. We annotate intermediate errors and correction behavior in the collected traces, including traces that end with a correct answer after correction. For traces with an incorrect final answer, we additionally trace back the cause of failure: we annotate the correctness of every intermediate tool output, record whether the agent attempts any verification or correction, and determine whether the root cause of the final error is uncorrected intermediate tool output propagation or tool orchestration failure (wrong tool selection or invocation order). Based on these annotations, we define two metrics: **Error Cascade Rate (ECR)**, the fraction of incorrect samples whose final error can be traced back to uncorrected intermediate tool output errors; and **Correction Rate (CR)**, the fraction of detected intermediate errors that the method successfully corrects, grouped by tool count ($\leq$3, 4–6, $>$6).

Results. Figure 2 reports the ECR and CR of each method, and Figure 3 illustrates typical error cascade instances under each paradigm. The analysis reveals three key findings.

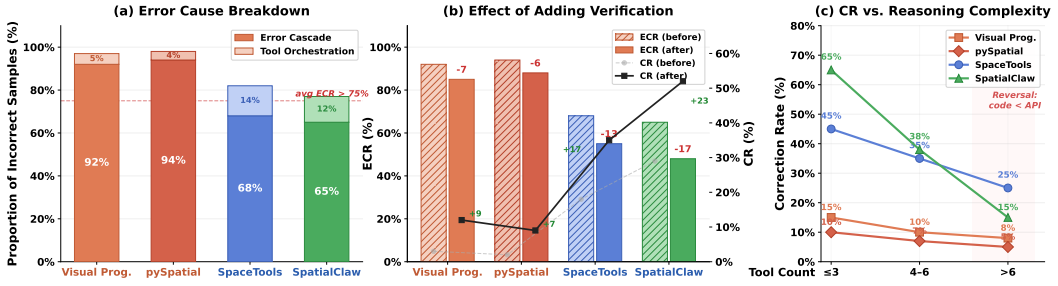

Figure 2: Empirical analysis of error cascade. (a) Error-cause breakdown across four methods. One-shot methods exceed 90%; step-by-step methods are lower but still above 60%. **(b)** ECR and CR before and after adding verification. All methods improve; step-by-step methods benefit significantly more, with the code-based method showing the largest gain. **(c)** Correction rate vs. tool count after adding verification. The code-based method achieves the highest CR ($\sim$65%) at $\leq$3 tools but plummets to $\sim$15% at >6 tools, falling below structured-API methods ($\sim$25%).

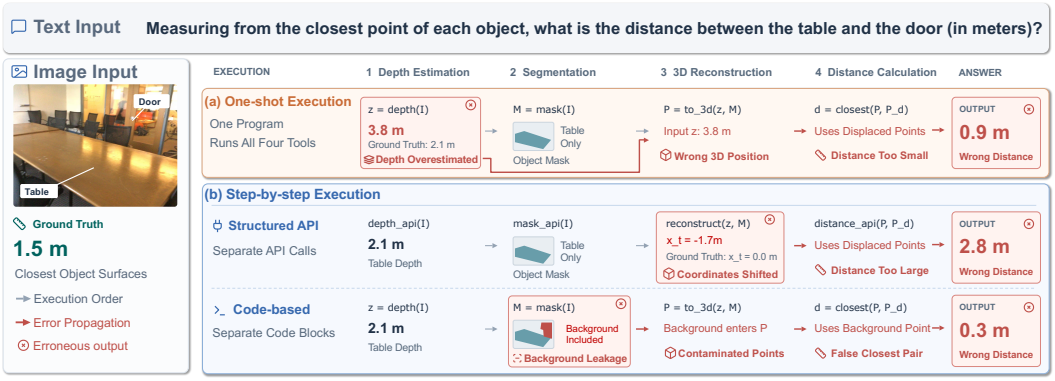

Figure 3: Error cascades from erroneous intermediate tool outputs. (a) One-shot: a depth error propagates through reconstruction to the final distance. **(b)** Step-by-step: errors originate in reconstruction (structured API) or segmentation (code-based), then propagate downstream.

Finding 1: Error cascade is the dominant cause of incorrect answers. As shown in Figure 2(a), the four ECRs are approximately 92%, 94%, 68%, and 65%, averaging 79.8%. The corresponding tool-orchestration shares are 5%, 4%, 14%, and 12%, averaging 8.8%. Error cascades thus account for more than 75% of failures on a method-average basis. These results identify insufficient quality control over intermediate outputs as the main accuracy bottleneck.

Finding 2: Step-by-step execution benefits more from verification. We augment each method with output checks and retry instructions supported by its execution interface. All methods achieve lower ECR and higher CR after verification (Figure 2(b)). Step-by-step methods benefit more because they can inspect and revise intermediate outputs, whereas one-shot methods can only inspect completed executions and regenerate or rerun the program. Within step-by-step execution, SpatialClaw benefits more than SpaceTools: arbitrary code enables customized verification and adjustment, while predefined APIs constrain this flexibility.

Finding 3: Complexity reverses the code-based correction advantage. Figure 2(c) groups post-verification CR by tool count. At $\leq$3 tools, code-based execution achieves 65% CR, exceeding the structured-API rate of 45%. At >6 tools, however, it falls to approximately 15%, below the structured-API rate of 25%. As reasoning chains span more tools and domains, documentation and execution history increase the context load and verification burden, leading to verification omission (skipped checks), tool misuse (incorrect parameters), and tool omission (missing useful calls). Structured-API methods degrade less steeply with shorter contexts, but their rigid interfaces still limit verification flexibility. None of the methods sustains effective correction under high complexity.

These findings highlight intermediate-output quality control as a key reliability bottleneck. Verification helps, but its benefits diminish with complexity. Suppressing error cascades therefore requires flexible, closed-loop verification at each execution step, and a manageable verification scope as tool count and domain complexity grow.

4 SPATIALCREW

The analysis in Section 3 motivates two requirements for reliable tool-augmented spatial reasoning: flexible correction of intermediate errors and a verification scope that remains manageable as reasoning spans multiple domains. We propose **SPATIALCREW**, a multi-agent collaboration framework, to address these requirements through domain-specialized agents, each operating in a **Write–Execute–Verify–Adjust (WEVA)** loop, and a coordinator that checks consistency across their outputs. Next, we introduce the overall workflow and tool organization, then describe agent responsibilities, local verification, and inter-agent communication.

4.1 OVERVIEW

As illustrated in Figure 1(c), SPATIALCREW comprises a Coordinator Agent and three domain specialists: the Perception, Geometry, and Temporal Agents. The Coordinator manages planning and cross-domain coordination, while the specialists handle visual evidence extraction, 3D spatial computation, and motion analysis, respectively, using domain-specific tools and verification criteria. Given images or video frames and a question, the Coordinator decomposes the task, identifies dependencies, and activates the required specialists. Each specialist executes its subtasks in a persistent Python kernel through a WEVA loop to check and refine intermediate outputs. The Coordinator forwards results to dependent subtasks, reviews outputs and verification logs for cross-domain consistency, requests refinement when needed, and synthesizes the final answer.

4.2 TOOL SET ORGANIZATION

We organize the tool set into three domain-specific toolkits (Table 1), reusing the perception and geometry capabilities of prior spatial reasoning agents (Cho et al., 2026; Chen et al., 2025a; Luo et al., 2026).

Perception extracts visual evidence, geometry converts it into spatial quantities, and temporal analysis relates observations across frames. All specialists can also use standard scientific computing libraries, including `numpy`, `scipy`, and `matplotlib`, for numerical computation and visualization.

The organizing principle is domain isolation: each specialist receives documentation only for its assigned toolkit, comprising two to four interfaces, with the corresponding verification rules. Thus, it can write custom processing and checking code without maintaining instructions for unrelated domains.

Table 1: Domain toolkits.

Agent	Tool	Function
Perception	`tools.DA3`	Depth estimation
	`tools.SAM3`	Segmentation
	`tools.Mask`	Mask Operations
	`tools.Draw`	Rendering
Geometry	`tools. Reconstruct`	Reconstruction
	`tools.Geometry`	Spatial Measures
Temporal	`tools.Time`	Motion Analysis
	`tools.Graph`	Scene Graphs

4.3 MULTI-AGENT ARCHITECTURE

The tool partition determines what each specialist computes and verifies, while the Coordinator maintains the dependencies between subtasks. Their complementary responsibilities are as follows.

Coordinator Agent. The Coordinator analyzes the question and constructs a dependency-aware plan, allowing independent subtasks to run in parallel. Each assignment includes the relevant visual input, available intermediate results, and an expected output specification describing the requested quantity and task-specific checks, such as target coverage for segmentation or a scene-appropriate geometric range. These specifications complement the specialists' generic domain rules. The Coordinator collects their outputs, performs cross-domain consistency checks, and integrates the evidence into the final answer. Its role is restricted to planning, dispatch, and verification.

Perception Agent. The Perception Agent estimates depth and segments objects, returning depth maps, masks, and bounding boxes with verification records. It checks depth positivity and distributional plausibility, screens segmentation masks using an area range of 0.1%–80% of the image, and visually

inspects the results through `show()`. Numerical checks and visual inspection provide complementary evidence about output quality. When a check fails, the agent revises segmentation prompts or depth-estimation parameters and re-executes the affected operation.

Geometry Agent. The Geometry Agent receives perception outputs through the Coordinator and performs 3D reconstruction, coordinate transformations, distance and angle computation, and plane fitting. It checks point-cloud density, transformation consistency, and the physical plausibility of computed quantities. Measurement strategies follow the requested spatial relation: centroid distances describe center separation, whereas KD-tree nearest-point search estimates closest separation. When verification exposes an unsuitable computation, the agent revises its strategy in accordance with that measurement definition and retries.

Temporal Agent. Activated for video or 4D tasks, the Temporal Agent uses frame sequences and available spatial outputs to establish correspondences, estimate velocities, construct trajectories, and analyze scene-graph evolution. Verification checks for implausible tracking discontinuities, agreement between motion estimates and the observed frame sequence, and event times within the video duration. Failed checks trigger revision of the temporal analysis before its results are returned.

4.4 Intra-Agent Write–Execute–Verify–Adjust Loop

The domain checks above are applied through a common execution loop. Each specialist interacts with a persistent Python kernel through four stages: ❶ Write, generating code from the assigned subtask and current state; ❷ Execute, running the code and observing variables, execution feedback, and visualizations; ❸ Verify, evaluating the outputs against both domain rules and the Coordinator's expected output specification; and ❹ when a check fails, Adjust, revising parameters, processing logic, or the computation strategy before re-execution. Each outcome is recorded with diagnostic information, making it available for subsequent refinement and coordination.

A successful check permits advancement to the next sub-step or return of the completed result; a failed check triggers revision before the affected output is used further. Kernel state persists across iterations, retaining loaded models, variables, and intermediate results so that refinement need not restart the entire computation. Execution is bounded by $N_{\max} = 30$ steps per specialist, consistent with Section 5. If the budget is exhausted, the agent returns its available result with low confidence and unresolved issues recorded. Local verification therefore provides a controlled refinement process, whose remaining uncertainties are handled through inter-agent communication.

4.5 Inter-Agent Communication

To support this coordination, specialists return a common four-field record: ❶ Result, containing computed outputs or references to Python variables in the shared namespace; ❷ Confidence, a self-assessed score in $[0, 1]$ based on verification outcomes; ❸ Verification_log, recording passed and failed checks with diagnostics; and ❹ Warnings, describing unresolved issues such as occlusion or sparse point clouds. The Coordinator forwards relevant results to downstream agents together with their subtask specifications, preserving the information needed to interpret intermediate outputs.

The Coordinator uses these records to check agreement across domains, for example whether a reconstructed distance is compatible with the depth evidence from which it was derived. Low confidence or conflicting outputs can trigger a request for additional verification by the relevant specialist. Confidence serves as a signal for review, while the verification logs provide the diagnostic basis for resolving discrepancies. This complements the WEVA loops: specialists inspect individual computations, and the Coordinator checks whether their outputs form a coherent basis for the answer.

4.6 System Prompt Design Principles

The preceding execution and communication rules are encoded in independent system prompts with four shared principles: ❶ Domain isolation, restricting each specialist's tool documentation and verification instructions to its own responsibilities; ❷ Mandatory verification, requiring checks after each execution step and refinement after a failed check; ❸ Explicit uncertainty, requiring confidence and warnings to reflect unresolved issues; and ❹ Cross-backbone consistency, using the same role-specific prompts across benchmarks and VLM backbones without task-specific or model-specific tuning. These instructions establish a protocol while keeping each specialist's context focused.

5 EXPERIMENTAL SETUP

We describe the benchmarks, metrics, baselines, and implementation before the evaluation.

Benchmarks. We evaluate on 20 benchmarks in five reporting groups: single-image spatial reasoning (4 benchmarks), multi-view spatial reasoning (3), general spatial reasoning (3), video spatial and 4D reasoning (6), and general visual understanding (4). The suite covers static, multi-view, and dynamic reasoning; Appendix A lists every benchmark and its group.

Task Metrics. We use accuracy for discrete answers and mean relative accuracy (MRA) for numerical estimates, both on a 0–100 scale. Category scores average their constituent benchmarks; Overall averages all 20 benchmarks with equal weights. Cross-backbone means weight each backbone equally. We report absolute score differences and percentage-point (pp) changes in rates; Appendix B.1 defines the aggregation protocol.

Reliability Metrics. Error cascade rate (ECR) is the percentage of incorrect final answers attributable to uncorrected intermediate tool errors, following Section 3. Correction success rate (CSR) is the percentage of error-containing traces in which correction leads to a correct final answer. Correction rate (CR) instead measures the fraction of detected intermediate errors that are corrected. Appendix B.1 specifies their distinct denominators and aggregation rules.

Baselines. With the primary backbone, we compare SPATIALCREW against six baselines. No-tool predicts directly from the visual inputs and question. Visual Programming (Gupta & Kembhavi, 2023) and pySpatial (Luo et al., 2026) use one-shot program execution. SpaceTools (Chen et al., 2025a) and GCA (Chen et al., 2025b) invoke structured APIs step by step, while SpatialClaw (Cho et al., 2026) executes code incrementally in a persistent Python environment.

Backbones. Gemma4-31B (Abd et al., 2026) is the primary backbone for the main results and ablations. We additionally evaluate Gemma4-26B-A4B, Qwen3.6-27B, Qwen3.6-35B-A3B, and Qwen3.5-122B-A10B (Qwen Team, 2026b;a), covering dense and mixture-of-experts (MoE) architectures. Role-specific prompts and tool configurations are shared across all five backbones.

Implementation. For benchmarks exceeding 1,000 examples, all methods use the same fixed random subset of 1,000. Each specialist uses at most $N_{\max} = 30$ reasoning steps, with tool access as in Table 1; decoding settings are fixed within each backbone comparison.

6 EXPERIMENTAL RESULTS

We evaluate error propagation, benchmark performance, cross-backbone generalization, component contributions, and inference efficiency.

6.1 ERROR CASCADE ANALYSIS

Table 2 shows that SPATIALCREW reduces ECR from the five tool-augmented baselines' mean of 78.0% to 23.4%, a reduction of 54.6 pp or 70.0% relative. No-tool is excluded because direct prediction has no intermediate tool errors. Against the strongest baseline, SpatialClaw, ECR decreases from 63.5% to 23.4%, an improvement of 40.1 pp. SPATIALCREW also achieves 64.2% CSR, exceeding SpatialClaw's 29.3% by 34.9 pp and restoring correct final answers in nearly two-thirds of error-containing traces. ECR and CSR condition on different sample sets and provide evidence of improved error containment and final-answer recovery.

Table 2: Intermediate reliability. Parentheses show Our SPATIALCREW minus each reference, in pp.

Method	ECR (%) ↓	CSR (%) ↑
Visual Prog.	91.3 (-67.9)	3.2 (+61.0)
pySpatial	93.7 (-70.3)	2.1 (+62.1)
SpaceTools	72.5 (-49.1)	18.4 (+45.8)
GCA	69.1 (-45.7)	21.3 (+42.9)
SpatialClaw	63.5 (-40.1)	29.3 (+34.9)
Baseline mean	78.0 (-54.6)	14.9 (+49.3)
Our SPATIALCREW	**23.4**	**64.2**

Table 3: Benchmark performance with Gemma4-31B. Scores are category macro-averages; Overall averages all 20 benchmarks. Green parentheses show the improvement of SPATIALCREW over each reference, computed before rounding. Best scores are bold.

Method	Single image	Multi view	General spatial	Video 4D	General visual	Overall
No-tool	55.6 (+7.9)	50.2 (+16.6)	62.4 (+3.8)	47.6 (+11.8)	55.5 (+5.2)	53.4 (+9.2)
Visual Prog.	56.2 (+7.3)	53.6 (+13.2)	61.6 (+4.6)	51.3 (+8.0)	56.3 (+4.4)	55.2 (+7.4)
pySpatial	53.2 (+10.3)	50.7 (+16.1)	56.0 (+10.2)	41.6 (+17.7)	53.3 (+7.5)	49.8 (+12.8)
SpaceTools	52.1 (+11.5)	46.6 (+20.2)	55.4 (+10.8)	44.1 (+15.3)	52.8 (+8.0)	49.5 (+13.1)
GCA	54.3 (+9.2)	48.2 (+18.6)	57.8 (+8.5)	43.5 (+15.9)	53.6 (+7.2)	50.5 (+12.1)
SpatialClaw	61.9 (+1.6)	62.5 (+4.3)	64.8 (+1.5)	55.1 (+4.2)	59.4 (+1.4)	59.9 (+2.7)
Baseline mean	55.5 (+8.0)	52.0 (+14.8)	59.7 (+6.6)	47.2 (+12.1)	55.1 (+5.6)	53.0 (+9.6)
Our SPATIALCREW	**63.5**	**66.8**	**66.2**	**59.3**	**60.8**	**62.6**

6.2 BENCHMARK PERFORMANCE

The reduction in error propagation is accompanied by consistent improvements in final task performance (Table 3). SPATIALCREW achieves an overall score of 62.6, compared with 53.0 averaged over all six baselines, a gain of 9.6 points. It also exceeds the strongest overall baseline, SpatialClaw, by 2.7 points (62.6 versus 59.9) and the no-tool backbone by 9.2 points. The complete results in Appendix A show that SPATIALCREW surpasses the strongest baseline on 19 of 20 benchmarks and ties it on ViewSpatial; it obtains the highest average in every reporting group.

The largest category gains over SpatialClaw occur on multi-view reasoning (+4.3) and video spatial and 4D reasoning (+4.2). These tasks require information to remain consistent across views, frames, and computational domains, creating more opportunities for a flawed intermediate result to affect later computations. Gains are smaller but remain positive for single-image reasoning (+1.6), general spatial reasoning (+1.5), and general visual understanding (+1.4). Meanwhile, pySpatial, SpaceTools, and GCA score below the no-tool baseline overall, showing that access to spatial tools alone does not guarantee better task performance.

6.3 CROSS-BACKBONE GENERALIZATION

Table 4: Cross-backbone generalization for SPATIALCREW. Acc is the benchmark macro-average used in Table 3; green parentheses give the gain over each corresponding no-tool backbone. Mean averages the four backbones shown. Best individual-backbone results are bold.

Metric	Gemma4 26B-A4B	Qwen3.6 27B	Qwen3.6 35B-A3B	Qwen3.5 122B-A10B	Mean
ECR (%) ↓	27.8	25.1	24.6	**22.8**	25.1
Acc ↑	57.1 (+9.1)	63.5 (+8.5)	64.0 (+9.5)	**67.5** (+9.0)	63.0 (+9.0)

The corresponding no-tool scores, in backbone column order, are 48.0, 55.0, 54.5, and 58.5.

All four additional backbones benefit from SPATIALCREW (Table 4), with score gains of 8.5–9.5 points over their corresponding no-tool models and ECRs of 22.8%–27.8%. Qwen3.5-122B-A10B attains the highest score of 67.5 and the lowest ECR of 22.8%, while Qwen3.6-35B-A3B achieves the largest score gain (+9.5). Including the primary Gemma4-31B configuration, the five-backbone mean score is 62.9, compared with 53.9 for the corresponding no-tool models, an average gain of 9.1 points; mean ECR is 24.7%. These improvements across dense and MoE backbones support the portability of the framework under shared prompts and tool configurations.

6.4 COMPONENT AND DOMAIN ANALYSIS

Table 5 isolates the contributions of verification and domain specialization. Removing intra-agent verification raises ECR from 23.4% to 41.7% and lowers the overall score by 3.5 points; removing

Table 5: Component ablations with Gemma4-31B. Parentheses report the ECR reduction or score improvement achieved by Our SPATIALCREW relative to each variant.

Variant	ECR (%) ↓	Overall ↑
Without intra-agent verification	41.7 (-18.3)	59.1 (+3.5)
Without cross-agent verification	29.6 (-6.2)	61.4 (+1.2)
Single agent, all tools	46.2 (-22.8)	58.5 (+4.1)
Single agent with verification	35.8 (-12.4)	60.2 (+2.4)
Our SPATIALCREW	**23.4**	**62.6**

cross-agent verification raises ECR by 6.2 pp and lowers the score by 1.2 points. Intra-agent verification has the larger effect, while the two components serve complementary roles: specialists check domain computations, and the Coordinator checks cross-domain consistency. A single agent with verification achieves 60.2 with 35.8% ECR, trailing SPATIALCREW by 2.4 score points and 12.4 ECR points. Without verification, its score decreases to 58.5 and ECR rises to 46.2%. Thus, verification improves a single agent, while distributing checks across specialists provides further gains. Appendix B.3 reports individual specialist ablations.

The domain-level comparison in Table 6 offers a more direct view of correction behavior. SPATIALCREW improves the correction rate in all three domains, with the largest absolute gain in temporal reasoning (+32.3 pp), followed by perception (+29.2 pp) and geometry (+27.2 pp). The advantage therefore extends beyond the initial perception stage to computations later in the reasoning chain. These domain-specific rates describe correction of intermediate errors; they should not be averaged to reconstruct CSR, which additionally requires a correct final answer and uses a sample-level denominator. Together with the component ablations, the results suggest that focused verification helps protect both foundational visual evidence and the geometric computations that depend on it.

Table 6: Domain-level correction rates (%).

Domain	SpatialClaw	Our SPATIALCREW
Perception	42.1 (+29.2)	**71.3**
Geometry	31.4 (+27.2)	**58.6**
Temporal	29.8 (+32.3)	**62.1**

6.5 INFERENCE EFFICIENCY

Table 7 summarizes the per-question runtime of SPATIALCREW by backbone and architecture, covering coordination, tool execution, and verification. Mean wall-clock times range from 42.6 to 68.4 seconds across the five backbones, with an equal-weight average of 55.5 seconds per question. Specialists are activated according to task requirements, keeping execution focused on the perception, geometry, and temporal operations relevant to each question. At approximately one minute per question on average, the framework offers an acceptable turnaround for spatial analysis and benchmark evaluation, where checking intermediate evidence is part of the reasoning process.

Table 7: Per-question runtime of Our SPATIALCREW.

Backbone	Type	Mean time (s/question)
Gemma4-31B	Dense	58.3
Gemma4-26B-A4B	MoE	42.6
Qwen3.6-27B	Dense	61.5
Qwen3.6-35B-A3B	MoE	46.8
Qwen3.5-122B-A10B	MoE	68.4
Mean (5 backbones)		**55.5**

7 CONCLUSION

In this paper, we first identified error cascades from unchecked intermediate tool outputs as a key reliability bottleneck and showed that verification becomes less effective as reasoning complexity grows. To address this, we proposed SPATIALCREW, combining domain-specific WEVA loops with cross-agent consistency checks. Experiments show consistent improvements in reliability and task performance. Our work advances more reliable spatial reasoning with effective tool orchestration.

AI USE STATEMENT

In this work, we used generative AI tools to assist with manuscript editing and language polishing, including improving grammar, clarity, and readability. Generative AI also assisted with the explicitly marked illustrative baseline estimates in Appendix B.2; these are not experimental measurements and are excluded from the reported results and conclusions. All AI-assisted revisions were reviewed and edited by the authors to preserve the intended scientific meaning and ensure consistency with the methods, experimental results, and conclusions. We take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

This work studies spatial reasoning using publicly available benchmarks. It does not introduce new personal-data collection. The framework's verification procedures improve reliability but do not guarantee that every spatial output is correct.

REPRODUCIBILITY STATEMENT

Section 5 describes the shared evaluation settings, Appendix A provides the full benchmark score matrix, and Appendix B records the metric definitions and implementation details. Appendix C summarizes the agent instructions. Code is available at `https://anonymous.4open.science/r/SpatialCrew-B650`.

REFERENCES

Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, and et al. Gemma 4 technical report. *CoRR*, abs/2607.02770, 2026. doi: 10.48550/ARXIV.2607.02770. URL `https://doi.org/10.48550/arXiv.2607.02770`.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Ming-Hsuan Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. *CoRR*, abs/2502.13923, 2025. doi: 10.48550/ARXIV.2502.13923. URL `https://doi.org/10.48550/arXiv.2502.13923`.

Boyuan Chen, Zhuo Xu, Sean Kirmani, Brian Ichter, Dorsa Sadigh, Leonidas J. Guibas, and Fei Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pp. 14455–14465. IEEE, 2024. doi: 10.1109/CVPR52733.2024.01370. URL `https://doi.org/10.1109/CVPR52733.2024.01370`.

Siyi Chen, Mikaela Angelina Uy, Chan Hee Song, Faisal Ladhak, Adithyavairavan Murali, Qing Qu, Stan Birchfield, Valts Blukis, and Jonathan Tremblay. Spacetools: Tool-augmented spatial reasoning via double interactive RL. *CoRR*, abs/2512.04069, 2025a. doi: 10.48550/ARXIV.2512.04069. URL `https://doi.org/10.48550/arXiv.2512.04069`.

Zeren Chen, Xiaoya Lu, Zhijie Zheng, Pengrui Li, Lehan He, Yijin Zhou, Jing Shao, Bohan Zhuang, and Lu Sheng. Geometrically-constrained agent for spatial reasoning. *CoRR*, abs/2511.22659, 2025b. doi: 10.48550/ARXIV.2511.22659. URL `https://doi.org/10.48550/arXiv.2511.22659`.

An-Chieh Cheng, Hongxu Yin, Yang Fu, Qiushan Guo, Ruihan Yang, Jan Kautz, Xiaolong Wang, and Sifei Liu. Spatialrgpt: Grounded spatial reasoning in vision-language models. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL `http://papers.nips.cc/paper_files/paper/2024/hash/f38cb4cf9a5eaa92b3cfa481832719c6-Abstract-Conference.html`.

Seokju Cho, Ryo Hachiuma, Abhishek Badki, Hang Su, Byung-Kwan Lee, Chan Hee Song, Sifei Liu, Subhashree Radhakrishnan, Seungryong Kim, Yu-Chiang Frank Wang, and Min-Hung Chen. Spatialclaw: Rethinking action interface for agentic spatial reasoning. *CoRR*, abs/2606.13673, 2026. doi: 10.48550/ARXIV.2606.13673. URL https://doi.org/10.48550/arXiv.2606.13673.

Zhiwen Fan, Jian Zhang, Renjie Li, Junge Zhang, Runjin Chen, Hezhen Hu, Kevin Wang, Huaizhi Qu, Dilin Wang, Zhicheng Yan, Hongyu Xu, Justin Theiss, Tianlong Chen, Jiachen Li, Zhengzhong Tu, Zhangyang Wang, and Rakesh Ranjan. VLM-3R: vision-language models augmented with instruction-aligned 3d reconstruction. *CoRR*, abs/2505.20279, 2025a. doi: 10.48550/ARXIV.2505.20279. URL https://doi.org/10.48550/arXiv.2505.20279.

Zhiwen Fan, Jian Zhang, Renjie Li, Junge Zhang, Runjin Chen, Hezhen Hu, Kevin Wang, Huaizhi Qu, Dilin Wang, Zhicheng Yan, Hongyu Xu, Justin Theiss, Tianlong Chen, Jiachen Li, Zhengzhong Tu, Zhangyang Wang, and Rakesh Ranjan. VLM-3R: vision-language models augmented with instruction-aligned 3d reconstruction. *CoRR*, abs/2505.20279, 2025b. doi: 10.48550/ARXIV.2505.20279. URL https://doi.org/10.48550/arXiv.2505.20279.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, Peixian Chen, Yanwei Li, Shaohui Lin, Sirui Zhao, Ke Li, Tong Xu, Xiawu Zheng, Enhong Chen, Caifeng Shan, Ran He, and Xing Sun. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pp. 24108–24118. Computer Vision Foundation / IEEE, 2025. doi: 10.1109/CVPR52734.2025.02245. URL https://openaccess.thecvf.com/content/CVPR2025/html/Fu_Video-MME_The_First-Ever_Comprehensive_Evaluation_Benchmark_of_Multi-modal_LLMs_in_CVPR_2025_paper.html.

Chaoyou Fu, Haozhi Yuan, Yuhao Dong, Yifan Zhang, Yunhang Shen, Xiaoxing Hu, Xueying Li, Jinsen Su, Chengwu Long, Xiaoyao Xie, Yongkang Xie, Xiawu Zheng, Xue Yang, Haoyu Cao, Yunsheng Wu, Ziwei Liu, Xing Sun, Caifeng Shan, and Ran He. Video-mme-v2: Towards the next stage in benchmarks for comprehensive video understanding. *CoRR*, abs/2604.05015, 2026. doi: 10.48550/ARXIV.2604.05015. URL https://doi.org/10.48550/arXiv.2604.05015.

Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A. Smith, Wei-Chiu Ma, and Ranjay Krishna. BLINK: multimodal large language models can see but not perceive. In *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part XXIII*, volume 15081 of *Lecture Notes in Computer Science*, pp. 148–166. Springer, 2024. doi: 10.1007/978-3-031-73337-6_9. URL https://doi.org/10.1007/978-3-031-73337-6_9.

Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pp. 14953–14962. IEEE, 2023. doi: 10.1109/CVPR52729.2023.01436. URL https://doi.org/10.1109/CVPR52729.2023.01436.

Mengdi Jia, Zekun Qi, Shaochen Zhang, Wenyao Zhang, Xinqiang Yu, Jiawei He, He Wang, and Li Yi. Omnispatial: Towards comprehensive spatial reasoning benchmark for vision language models. volume abs/2506.03135, 2025. doi: 10.48550/ARXIV.2506.03135. URL https://doi.org/10.48550/arXiv.2506.03135.

Dingming Li, Hongxing Li, Zixuan Wang, Yuchen Yan, Hang Zhang, Siqi Chen, Guiyang Hou, Shengpei Jiang, Wenqiao Zhang, Yongliang Shen, Weiming Lu, and Yueting Zhuang. Viewspatial-bench: Evaluating multi-perspective spatial localization in vision-language models. *CoRR*, abs/2505.21500, 2025a. doi: 10.48550/ARXIV.2505.21500. URL https://doi.org/10.48550/arXiv.2505.21500.

Hongxing Li, Dingming Li, Zixuan Wang, Yuchen Yan, Hang Wu, Wenqi Zhang, Yongliang Shen, Weiming Lu, Jun Xiao, and Yueting Zhuang. Spatialladder: Progressive training for spatial

reasoning in vision-language models. *CoRR*, abs/2510.08531, 2025b. doi: 10.48550/ARXIV.2510.08531. URL https://doi.org/10.48550/arXiv.2510.08531.

Shaoxuan Li, Zhixuan Zhao, Hanze Deng, Zirun Ma, Shulin Tian, Zuyan Liu, Yushi Hu, Haoning Wu, Yuhao Dong, Benlin Liu, Ziwei Liu, and Ranjay Krishna. Perceptioncomp: A video benchmark for complex perception-centric reasoning. *CoRR*, abs/2603.26653, 2026. doi: 10.48550/ARXIV.2603.26653. URL https://doi.org/10.48550/arXiv.2603.26653.

Jingli Lin, Runsen Xu, Shaohao Zhu, Sihan Yang, Peizhou Cao, Yunlong Ran, Miao Hu, Chenming Zhu, Yiman Xie, Yilin Long, Wenbo Hu, Dahua Lin, Tai Wang, and Jiangmiao Pang. Mmsi-video-bench: A holistic benchmark for video-based spatial intelligence. *CoRR*, abs/2512.10863, 2025. doi: 10.48550/ARXIV.2512.10863. URL https://doi.org/10.48550/arXiv.2512.10863.

Zhanpeng Luo, Ce Zhang, Silong Yong, Cunxi Dai, Qianwei Wang, Haoxi Ran, Guanya Shi, Katia P. Sycara, and Yaqi Xie. pyspatial: Generating 3d visual programs for zero-shot spatial reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=yv15C8ql24.

Damiano Marsili, Rohun Agrawal, Yisong Yue, and Georgia Gkioxari. Visual agentic AI for spatial reasoning with a dynamic API. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pp. 19446–19455. Computer Vision Foundation / IEEE, 2025. doi: 10.1109/CVPR52734.2025.01811. URL https://openaccess.thecvf.com/content/CVPR2025/html/Marsili_Visual_Agentic_AI_for_Spatial_Reasoning_with_a_Dynamic_API_CVPR_2025_paper.html.

Qwen Team. Qwen3.5-122B-A10B model card. https://huggingface.co/Qwen/Qwen3.5-122B-A10B, 2026a.

Qwen Team. Qwen3.6 model cards. https://huggingface.co/collections/Qwen/qwen36, 2026b.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloé Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross B. Girshick, Piotr Dollár, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=Ha6RTeWMd0.

Chan Hee Song, Valts Blukis, Jonathan Tremblay, Stephen Tyree, Yu Su, and Stan Birchfield. Robospatial: Teaching spatial understanding to 2d and 3d vision-language models for robotics. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pp. 15768–15780. Computer Vision Foundation / IEEE, 2025. doi: 10.1109/CVPR52734.2025.01470. URL https://openaccess.thecvf.com/content/CVPR2025/html/Song_RoboSpatial_Teaching_Spatial_Understanding_to_2D_and_3D_Vision-Language_Models_CVPR_2025_paper.html.

Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pp. 11854–11864. IEEE, 2023. doi: 10.1109/ICCV51070.2023.01092. URL https://doi.org/10.1109/ICCV51070.2023.01092.

Gemini Robotics Team. Gemini robotics: Bringing AI into the physical world. *CoRR*, abs/2503.20020, 2025. doi: 10.48550/ARXIV.2503.20020. URL https://doi.org/10.48550/arXiv.2503.20020.

Mingrui Wu, Zhaozhi Wang, Fangjinhua Wang, Jiaolong Yang, Marc Pollefeys, and Tong Zhang. From indoor to open world: Revealing the spatial reasoning gap in mllms. *CoRR*, abs/2512.19683, 2025. doi: 10.48550/ARXIV.2512.19683. URL https://doi.org/10.48550/arXiv.2512.19683.

Yuxi Xiao, Longfei Li, Shen Yan, Xinhang Liu, Sida Peng, Yunchao Wei, Xiaowei Zhou, and Bingyi Kang. Spatialtree: How spatial abilities branch out in mllms. *CoRR*, abs/2512.20617, 2025. doi: 10.48550/ARXIV.2512.20617. URL https://doi.org/10.48550/arXiv.2512.20617.

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pp. 10632–10643. Computer Vision Foundation / IEEE, 2025a. doi: 10.1109/CVPR52734.2025.00994. URL https://openaccess.thecvf.com/content/CVPR2025/html/Yang_Thinking_in_Space_How_Multimodal_Large_Language_Models_See_Remember_CVPR_2025_paper.html.

Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pp. 10371–10381. IEEE, 2024. doi: 10.1109/CVPR52733.2024.00987. URL https://doi.org/10.1109/CVPR52733.2024.00987.

Sihan Yang, Runsen Xu, Yiman Xie, Sizhe Yang, Mo Li, Jingli Lin, Chenming Zhu, Xiaochen Chen, Haodong Duan, Xiangyu Yue, Dahua Lin, Tai Wang, and Jiangmiao Pang. Mmsi-bench: A benchmark for multi-image spatial intelligence. volume abs/2505.23764, 2025b. doi: 10.48550/ARXIV.2505.23764. URL https://doi.org/10.48550/arXiv.2505.23764.

Baiqiao Yin, Qineng Wang, Pingyue Zhang, Jianshu Zhang, Kangrui Wang, Zihan Wang, Jieyu Zhang, Keshigeyan Chandrasegaran, Han Liu, Ranjay Krishna, Saining Xie, Manling Li, Jiajun Wu, and Li Fei-Fei. Spatial mental modeling from limited views. *CoRR*, abs/2506.21458, 2025. doi: 10.48550/ARXIV.2506.21458. URL https://doi.org/10.48550/arXiv.2506.21458.

Jiahui Zhang, Yurui Chen, Yueming Xu, Ze Huang, Jilin Mei, Junhui Chen, Yanpeng Zhou, Yu-Jie Yuan, Xinyue Cai, Guowei Huang, Xingyue Quan, Hang Xu, and Li Zhang. From flatland to space: Teaching vision-language models to perceive and reason in 3d. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025a. URL http://papers.nips.cc/paper_files/paper/2025/hash/e3a0db7c0a191854c176af1d20cdec80-Abstract-Datasets_and_Benchmarks_Track.html.

Ziang Zhang, Zehan Wang, Guanghao Zhang, Weilong Dai, Yan Xia, Ziang Yan, Minjie Hong, and Zhou Zhao. Dsi-bench: A benchmark for dynamic spatial intelligence. *CoRR*, abs/2510.18873, 2025b. doi: 10.48550/ARXIV.2510.18873. URL https://doi.org/10.48550/arXiv.2510.18873.

Fengzhe Zhou, Jiannan Huang, Jialuo Li, Deva Ramanan, and Humphrey Shi. Pai-bench: A comprehensive benchmark for physical AI. *CoRR*, abs/2512.01989, 2025. doi: 10.48550/ARXIV.2512.01989. URL https://doi.org/10.48550/arXiv.2512.01989.

Nannan Zhu, Yonghao Dong, Teng Wang, Xueqian Li, Shengjun Deng, Yijia Wang, Zheng Hong, Tiantian Geng, Guo Niu, Hanyan Huang, et al. Cvbench: Benchmarking cross-video synergies for complex multimodal reasoning. *arXiv preprint arXiv:2508.19542*, 2025.

APPENDIX

APPENDIX A FULL BENCHMARK RESULTS

Table 8 reports all 20 benchmark scores with Gemma4-31B. Each benchmark belongs to exactly one reporting group: single-image reasoning includes ERQA, Omni3D-Bench, OmniSpatial, and SPBench; multi-view reasoning includes MindCube, MMSI, and SPAR-Bench; general spatial reasoning includes BLINK, SpatialTree, and ViewSpatial; video spatial and 4D reasoning includes MMSI-Video, OSI-Bench, PAI-Bench, VSI-Bench-U, VSTI-Bench, and DSI-Bench; and general visual understanding includes CVBench, PerceptionComp, Video-MME, and Video-MME-v2.

Table 8: Full benchmark results. Each row uses the benchmark's task score (accuracy or MRA). Green parentheses show Our SPATIALCREW minus each baseline in score points. Bold indicates the best score, including ties. Overall is the unweighted mean of all 20 rows.

Benchmark	No-tool	Visual Prog.	pySpatial	SpaceTools	GCA	SpatialClaw	Our SPATIALCREW
Single-image spatial reasoning							
ERQA (2025)	58.3 (+4.7)	58.3 (+4.7)	50.8 (+12.2)	52.0 (+11.0)	53.5 (+9.5)	61.3 (+1.7)	**63.0**
Omni3D-Bench (2025)	51.7 (+5.1)	48.2 (+8.6)	50.6 (+6.2)	50.7 (+6.1)	51.2 (+5.6)	54.3 (+2.5)	**56.8**
OmniSpatial (2025)	57.3 (+7.9)	60.0 (+5.2)	58.0 (+7.2)	60.4 (+4.8)	57.5 (+7.7)	63.6 (+1.6)	**65.2**
SPBench (2025b)	55.1 (+13.9)	58.2 (+10.8)	53.5 (+15.5)	45.1 (+23.9)	55.0 (+14.0)	68.4 (+0.6)	**69.0**
Multi-view spatial reasoning							
MindCube (2025)	57.5 (+17.7)	57.5 (+17.7)	67.1 (+8.1)	52.9 (+22.3)	50.3 (+24.9)	72.8 (+2.4)	**75.2**
MMSI (2025b)	37.9 (+18.8)	42.3 (+14.4)	33.2 (+23.5)	33.1 (+23.6)	38.5 (+18.2)	51.3 (+5.4)	**56.7**
SPAR-Bench (2025a)	55.2 (+13.3)	61.1 (+7.4)	51.7 (+16.8)	53.9 (+14.6)	55.8 (+12.7)	63.3 (+5.2)	**68.5**
General spatial reasoning							
BLINK (2024)	75.7 (+0.8)	73.9 (+2.6)	60.3 (+16.2)	58.2 (+18.3)	62.0 (+14.5)	73.4 (+3.1)	**76.5**
SpatialTree (2025)	59.9 (+2.1)	58.9 (+3.1)	55.0 (+7.0)	52.5 (+9.5)	54.8 (+7.2)	60.7 (+1.3)	**62.0**
ViewSpatial (2025a)	51.7 (+8.5)	52.0 (+8.2)	52.8 (+7.4)	55.5 (+4.7)	56.5 (+3.7)	**60.2** (+0.0)	**60.2**
Video spatial and 4D reasoning							
MMSI-Video (2025)	36.9 (+9.3)	37.1 (+9.1)	28.7 (+17.5)	36.6 (+9.6)	33.8 (+12.4)	41.6 (+4.6)	**46.2**
OSI-Bench (2025)	35.6 (+10.9)	36.8 (+9.7)	32.0 (+14.5)	30.2 (+16.3)	31.5 (+15.0)	41.9 (+4.6)	**46.5**
PAI-Bench (2025)	65.0 (+6.0)	64.2 (+6.8)	46.0 (+25.0)	65.1 (+5.9)	62.8 (+8.2)	68.1 (+2.9)	**71.0**
VSI-Bench-U (2025a)	48.0 (+5.0)	47.8 (+5.2)	44.4 (+8.6)	33.6 (+19.4)	35.2 (+17.8)	48.5 (+4.5)	**53.0**
VSTI-Bench (2025b)	54.7 (+16.1)	64.2 (+6.6)	55.0 (+15.8)	56.0 (+14.8)	55.5 (+15.3)	67.6 (+3.2)	**70.8**
DSI-Bench (2025b)	45.3 (+23.2)	57.9 (+10.6)	43.7 (+24.8)	43.0 (+25.5)	42.0 (+26.5)	62.9 (+5.6)	**68.5**
General visual understanding							
CVBench (2025)	69.8 (+3.7)	71.3 (+2.2)	68.5 (+5.0)	67.8 (+5.7)	68.5 (+5.0)	72.2 (+1.3)	**73.5**
PerceptionComp (2026)	36.8 (+10.0)	39.2 (+7.6)	35.5 (+11.3)	34.8 (+12.0)	36.2 (+10.6)	44.0 (+2.8)	**46.8**
Video-MME (2025)	74.8 (+2.7)	74.6 (+2.9)	73.0 (+4.5)	72.5 (+5.0)	73.5 (+4.0)	77.0 (+0.5)	**77.5**
Video-MME-v2 (2026)	40.7 (+4.5)	40.2 (+5.0)	36.0 (+9.2)	36.0 (+9.2)	36.2 (+9.0)	44.4 (+0.8)	**45.2**
Overall	53.4 (+9.2)	55.2 (+7.4)	49.8 (+12.8)	49.5 (+13.1)	50.5 (+12.1)	59.9 (+2.7)	**62.6**

APPENDIX B SUPPLEMENTARY EXPERIMENTAL DETAILS

This appendix specifies the aggregation and trace metrics used in the main text, then reports specialist-removal and correction-trigger statistics. All comparisons use the same primary backbone unless another model is explicitly identified.

B.1 SCORE AGGREGATION AND METRIC DENOMINATORS

For method m, let $s_{m,b}$ be its score on benchmark b, and let $\mathcal{B}_k$ contain the benchmarks assigned to group k. We compute $S_m = \frac{1}{20}\sum_{b=1}^{20} s_{m,b}$ and $S_{m,k} = \frac{1}{|\mathcal{B}_k|}\sum_{b\in\mathcal{B}_k} s_{m,b}$. The groups contain 4, 3, 3, 6, and 4 benchmarks, respectively, so averaging the five category scores with equal weights would yield a different statistic. The displayed per-benchmark values sum to 1,252.1 for Our SPATIALCREW and 1,197.5 for SpatialClaw, giving means of 62.605 and 59.875 and a difference of 2.730 before rounding. We calculate every category mean, overall mean, and gap from the displayed benchmark rows, and round only the final value to one decimal place. Consequently, a gap may differ by 0.1 from subtracting two already-rounded category scores.

The mean over all six baseline overall scores is 53.0425, yielding a 9.5625-point improvement for Our SPATIALCREW before rounding. For the ECR comparison, only the five tool-augmented baselines are eligible: their reported rates sum to 390.1 and average 78.02%. Our SPATIALCREW therefore reduces ECR by 54.62 pp, or approximately 70.0% relative to this baseline mean. No-tool has no intermediate tool trace and is not assigned an ECR or CSR of zero.

Cross-backbone statistics give equal weight to each backbone. The four additional models in Table 4 average 63.025 in task score and 54.0 without tools, giving a 9.025-point gain and a mean ECR of 25.075%. Including Gemma4-31B, whose unrounded task scores are 62.605 for Our SPATIALCREW and 53.395 without tools, gives five-backbone means of 62.941 and 53.879, a gain of 9.062 points, and a mean ECR of 24.74%. These become 62.9, 53.9, +9.1, and 24.7%, respectively, after rounding. The 78.02% tool-augmented baseline mean above refers to the Gemma4-31B comparison.

ECR conditions on an incorrect final answer: $\mathrm{ECR} = 100N_{\mathrm{cascade}}/N_{\mathrm{wrong}}$. CSR instead conditions on the presence of an intermediate tool error: $\mathrm{CSR} = 100N_{\mathrm{corrected\ and\ correct}}/N_{\mathrm{error}}$. Here, the CSR numerator counts traces in which the relevant error is corrected and the final answer is correct. CR is an error-level diagnostic, $\mathrm{CR} = 100n_{\mathrm{corrected}}/n_{\mathrm{detected}}$. A zero denominator makes a rate undefined rather than zero. The full-suite ECR and CSR are reported benchmark averages over eligible trace sets, whereas Figure 2 uses the 300-question diagnostic subset. The four original-method ECRs displayed in that figure average 79.75% (79.8% rounded); this is a separate comparison from the five-baseline, full-suite mean of 78.02%. Domain-specific correction rates use their corresponding domain error sets and are not interchangeable with final-answer CSR.

B.2 CROSS-BACKBONE BASELINE COMPARISON: ILLUSTRATIVE ESTIMATES

Table 9 lays out a supplementary comparison with SpaceTools (structured-API execution) and SpatialClaw (code-based execution).

Table 9: Cross-backbone comparison layout with illustrative baseline estimates..

Method	Metric	Gemma4 26B-A4B	Qwen3.6 27B	Qwen3.6 35B-A3B	Qwen3.5 122B-A10B	Mean
SpaceTools	ECR ↓	76.9	74.2	73.7	71.9	74.2
	Acc ↑	44.0	50.4	50.9	54.4	49.9
SpatialClaw	ECR ↓	67.9	65.2	64.7	62.9	65.2
	Acc ↑	54.4	60.8	61.3	64.8	60.3
Our SPATIALCREW	ECR ↓	27.8	25.1	24.6	22.8	25.1
	Acc ↑	57.1	63.5	64.0	67.5	63.0

B.3 SPECIALIST CONTRIBUTIONS AND CORRECTION ACTIVITY

The specialist-removal study in Table 10 removes one specialist at a time and routes its computations to the Coordinator. This is an ablation-only exception to the full system's restriction on direct Coordinator tool use. The largest score decrease follows removal of the Perception Agent (2.9 points), consistent with the dependence of later computations on reliable depth and segmentation. Removing the Geometry Agent reduces the score by 2.2 points, while removing the Temporal Agent reduces it by 0.5 points. The smaller suite-wide temporal effect reflects its selective activation on temporal tasks and should not be interpreted as its effect on the video subset alone. All score gaps use the unrounded full-system mean of 62.605.

Table 10: Specialist-removal ablation.

Variant	Overall ↑
Without Perception	59.7 (+2.9)
Without Geometry	60.4 (+2.2)
Without Temporal	62.1 (+0.5)
Our SPATIALCREW	**62.6**

The verification logs provide a complementary measure of correction activity. At least one Perception Agent correction occurs in 34.2% of samples, and at least one Geometry Agent correction in 22.8%. For the Temporal Agent, the corresponding rate is 41.5% among the video samples where it is activated. These are trigger frequencies, not correction success rates, and the temporal denominator differs from the other two. They describe how often refinement is used; they do not, by themselves, establish a causal accuracy gain from correction.

B.4 Implementation and Runtime Details

Specialists use persistent Python state and the domain interfaces listed in Table 1. The `tools.DA3` and `tools.SAM3` interfaces provide depth estimation and segmentation, while the geometry and temporal interfaces operate on the resulting spatial evidence. Each specialist has a budget of $N_{\max} = 30$ reasoning steps. Role-specific prompts and tool configurations remain shared across the five backbones, with decoding settings fixed within each backbone comparison.

Runtime is defined as end-to-end wall-clock time from receipt of the prepared input to production of the final answer, including coordination, domain-tool execution, and verification. Table 7 gives a per-question mean for each backbone; Gemma4-31B averages 58.3 seconds over 1,000 questions. The five backbone means sum to 277.6 seconds, yielding an equal-weight mean of 55.52 seconds, displayed as 55.5 seconds per question. This statistic averages the per-backbone means, rather than pooling individual queries across backbones.

Appendix C Agent Prompt Summaries

The following summaries collect the instruction structure used by the four agents. They explain the division of responsibilities and verification expectations, rather than presenting complete executable system prompts. All specialists follow the same WEVA protocol and return results with confidence, a verification log, and warnings; the domain-specific instructions determine what is checked and how failed checks are handled.

Coordinator. The Coordinator identifies the required domains, decomposes the question into dependent subtasks, and assigns each subtask with the relevant inputs and an expected output specification. It forwards verified evidence between specialists, examines cross-domain consistency, and requests refinement when results conflict or unresolved warnings affect the answer. In the full framework it does not invoke domain tools directly; it synthesizes the final answer from the returned evidence and verification records.

Perception. The Perception Agent uses depth, segmentation, mask-processing, and visualization interfaces to extract visual evidence. Its checks include positive and plausible depth values, a segmentation-mask area between 0.1% and 80% of the image, and visual inspection through `show()`. The mask-area range is a screening rule rather than proof that a mask identifies the correct object. If a check fails, the agent revises the prompt, parameters, or processing logic before returning the outputs with any unresolved uncertainty.

Geometry. The Geometry Agent operates on the relevant visual evidence and performs reconstruction, coordinate transformations, and spatial measurements. It checks consistency of frames and

units, point-cloud support, and the plausibility of geometric outputs. The requested spatial relation determines the computation: center separation and closest-surface distance require different measurements. A failed check prompts revision of the affected computation, while uncertainty inherited from perception is retained in the returned record.

Temporal. The Temporal Agent relates observations across frames to estimate correspondence, motion, trajectories, and event timing. It checks temporal ordering, motion continuity, compatibility with the observed frame sequence, and event times within the video duration. Discontinuities or conflicting evidence trigger revision of the relevant temporal analysis. As for the other specialists, exhaustion of the $N_{\max} = 30$ step budget leads to a return with unresolved issues recorded, rather than treating unchecked outputs as verified.