S-AGENT 2.0: SPATIAL REASONING WITH A GENERAL CODING AGENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Spatial reasoning asks a model to work out where objects are in 3D space, how they relate to one another, and how they move. Agentic VLMs have recently brought large improvements on such tasks by letting the model actively gather evidence with 3D perception tools instead of answering in a single pass. Even so, on many questions the tools return the right evidence and the model still does not arrive at the correct answer: the gap between model and tools, where raw readings become usable conclusions, is left unfilled. We present S-Agent 2.0, a spatial reasoning agent that equips an off-the-shelf *general coding agent* with a command-line perception toolkit, and we design the interface around one principle: help the VLM *close the evidence loop—acquire* scene evidence, *remember* it in a persistent scene memory, and *interpret* each reading in plain spatial language. This design keeps the full flexibility of a coding agent, and every reading it acts on stays grounded in the tools' geometry. On MMSI-Bench and ReVSI, the toolkit improves a strong open-source model, Qwen3.8-27B, by $+13.9$ and $+20.5$ points over the same agent without tools, and a strong closed-source model, Gemini-3.8-Flash, by $+11.2$ and $+9.6$ points. With the open backbone S-Agent 2.0 reaches 66.5 on MMSI-Bench and 82.3 on ReVSI, ahead of the specialized system SpatialClaw on the same backbone (62.6 and 74.5), and with Gemini-3.8-Flash it reaches 74.1 and 90.5 against its 67.0 and 79.7. Both tool gains shrink as the backbone grows more capable, but SpatialClaw's shrinks far more on ReVSI, from $+13.2$ to $+2.7$, while ours stays at $+9.6$. S-Agent 2.0 also works across different general coding agent harnesses, such as Codex and Claude Code.

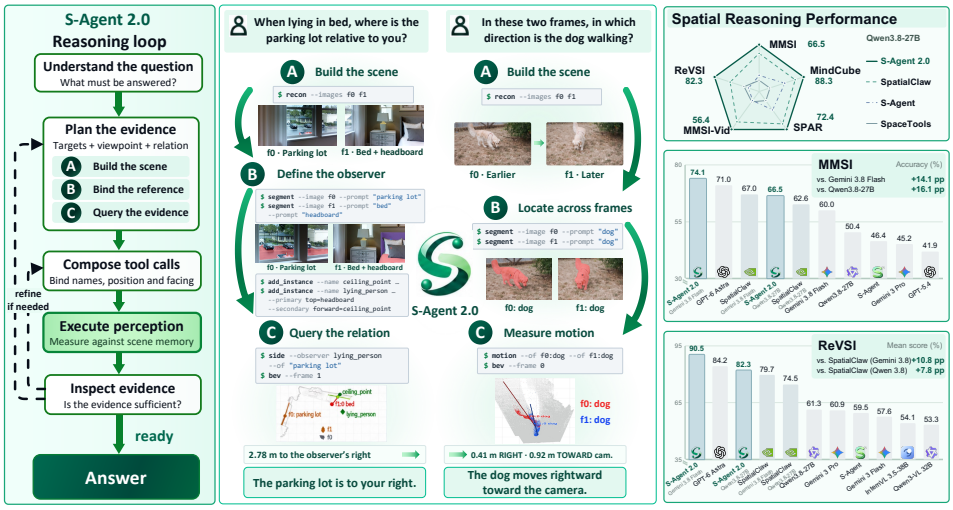


Figure 1: S-Agent 2.0 is a spatial reasoning agent built on a general coding agent. A command-line perception toolkit lets the agent close the evidence loop, acquiring scene evidence, remembering it in a persistent scene memory, and interpreting every reading in plain spatial language. S-Agent 2.0 delivers strong performance: it leads MMSI-Bench and ReVSI and stays ahead of the specialized agentic system SpatialClaw on every benchmark and backbone we test.

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.

1 INTRODUCTION

Spatial intelligence, the ability to understand where things are, how they relate in three dimensions, and how the world would look as one moves through it, is foundational to visual perception (Chen et al., 2024; Yang et al., 2025) and essential for any agent that operates in the physical world (Driess et al., 2023; Brohan et al., 2023; Song et al., 2025; Geiger et al., 2012; Caesar et al., 2020). Agentic VLMs such as S-Agent (Dai et al., 2026) have recently brought large improvements here: instead of answering in a single pass, the model actively gathers evidence with 3D perception tools while it reasons (Luo et al., 2026; Chen et al., 2026a). More recent work such as SpatialClaw (Cho et al., 2026) responds with a specialized harness: a custom state machine that drives the VLM through a hand-designed loop (plan, act, reflect) over a persistent code kernel with a library of 3D perception and geometry utilities, leveraging the flexibility of code to bridge the gap. Even so, on many questions the tools return the right evidence and the model still does not arrive at the correct answer (see Figure 4).

Watching such a system work reveals where it actually fails. A spatial answer is grounded only when a complete **evidence loop** closes: *acquire* the measurement, *remember* it while the rest of the evidence is gathered, and *interpret* the reading into understanding. Specialized systems invest almost entirely in acquisition, in the form of ever richer perception and geometry libraries, and leave the other two steps to the model. In our analysis of SpatialClaw trajectories, acquisition is rarely what breaks. The failures come in two forms. In the first, the model misreads a reading it holds: it flips the sign of a rotation it measured correctly, or reads a correct number in the wrong coordinate frame or the wrong sense. In the second, the evidence never reaches the answer at all: the measurement sits in the workspace while the model eyeballs the pictures, or a correct reading is discarded for an unrelated stated reason. The second form exposes a break between collecting evidence and drawing the conclusion. Once the readings are in, the reasoning that produces the answer runs detached from them, and the model falls back on its own perception of the scene. On the surface only interpretation goes wrong; underneath, the evidence that was acquired is never fully used. Closing the loop therefore places a requirement on each step. Acquisition has to combine easily with the VLM's own perception, so that what the model recognizes and what the tools measure enter the same scene. Acquired evidence needs a memory that holds it without effort, so that it stays within reach until the answer is written. And interpretation needs spatial language, so that a reading arrives in the terms the question is asked in.

1.1 S-AGENT 2.0

S-Agent 2.0 is our answer to those requirements: a spatial reasoning agent assembled from a stock general-purpose coding agent (OpenAI, 2025; Anthropic, 2025; Wang et al., 2025c) and a command-line perception toolkit designed to meet them. General-purpose coding agents (the harnesses behind modern software-engineering agents) already provide the scaffolding a spatial reasoner needs (a shell, a file system, code execution, long-horizon context management, a mature tool-calling loop), and two of their habits bear directly on the requirements above. First, looking at a frame or at a segmentation mask is an ordinary file view in the agent's stream, so the model's own look at the scene and the tools' readouts interleave in one conversation, with no separate image channel between them.(Yang et al., 2024) Second, a coding agent inspects and revises by reflex: it views an output, corrects what is wrong, and runs again. (Yang et al., 2024; Gou et al., 2024; Shinn et al., 2023) Applied to a scene, that reflex is what joins the two kinds of perception, since the model's own perception is checked against what the tools registered before any reading is taken. What these agents lack is perception itself; we keep the harness stock and add it as commands. The design effort goes into the toolkit, which meets the three requirements one by one.

Concretely, (i) acquisition combines two kinds of perception, the tools' measurements and the model's own: the model can register what it perceives or imagines as instances in the scene and drop what the tools got wrong, so both kinds land on one geometric footing; (ii) everything measured persists in a scene memory that later steps query by name instead of re-perceiving, so acquired evidence stays within reach until the answer is written; and (iii) every measurement comes back as an answer-ready reading stated in plain spatial language with its interpretation attached.

Our contributions can be summarized as follows:

- We build a spatial reasoning agent from a stock general-purpose coding agent and a command-line perception toolkit, and show that the method works across different general coding agent harnesses (§3.3).
- We design a command-line perception toolkit that closes the evidence loop for the agent: each call acquires scene evidence, writes it into a persistent scene memory, and returns its reading already interpreted in spatial language, and the model's own perception is grounded in the same scene (§2).
- S-Agent 2.0 delivers strong performance. On MMSI-Bench and ReVSI, the perception toolkit adds $+13.9$ and $+20.5$ points over the same agent without tools, on the already strong Qwen3.8-27B backbone (§3.2), and S-Agent 2.0 surpasses the state-of-the-art agentic spatial reasoning method, whose tool gain on ReVSI shrinks far more on the stronger frontier model Gemini-3.8-Flash while ours remains substantial (§3.3).

1.2 TAKEAWAYS

1. **The method works across coding-agent harnesses.** S-Agent 2.0 works across coding-agent harnesses, including OpenHands (Wang et al., 2025c), Codex (OpenAI, 2025) and Claude Code (Anthropic, 2025) (§3.3).
2. **The coding agent's inspect-and-revise habit joins the model's perception with the tools'.** The model registers what it perceives or imagines into the scene memory, drops what the tools got wrong (§2.3), and cross-checks the memory against the perception outputs (§3.4).
3. **The remaining gap is model-to-tool grounding.** The model's reasoning is mostly right; what remains is grounding the question into a measurement, which fails in two classes (§3.4).

2 METHOD

The diagnosis of §1 turns into three requirements that a system must meet to close the evidence loop, beyond merely possessing tools. *Acquisition joined with the model's own perception*: what the model recognizes and what the tools measure must enter one scene. *Persistent memory*: evidence gathered at one step must remain intact and addressable many steps later, without the model carrying it through its own context. *Interpretation on arrival*: a measurement should arrive in the spatial language the question is asked in. Existing systems supply acquisition generously, offering a Turing-complete kernel over perception and geometry libraries, but they keep the model's own perception in a separate channel and leave memory and interpretation to the model. S-Agent 2.0 inverts that allocation: we use a general coding agent as the spatial agent, and the design effort moves into a tool interface that joins the two kinds of perception and carries memory and interpretation itself. Figure 2 sketches the overall framework.

2.1 A GENERAL CODING AGENT AS THE SPATIAL AGENT

S-Agent 2.0 has three parts: a general-purpose coding agent π as planner and executor; a command-line interface (CLI) perception toolkit $\mathcal{T}$; and a persistent per-episode *scene memory* S that the toolkit reads and writes. Formally, a spatial question consists of a question q and frames $\mathcal{I} = \{I_1, \ldots, I_n\}$. The agent receives a fixed instruction u, the same for every episode, which holds a tutorial for the toolkit's commands and a recommended workflow: decide the evidence $\mathcal{E}$ the question requires, acquire $\mathcal{E}$ through the toolkit, check the perception tool outputs, and answer from the acquired readings. At reasoning step t the agent's context holds the instruction u, the question q, and the history of its own chain of thought $z_{<t}$, actions $a_{<t}$, and tool responses $o_{<t}$; from this context the policy π produces the next thought and action,

$$(z_t,\, a_t) \sim \pi(\cdot \mid u,\, q,\, z_{<t},\, a_{<t},\, o_{<t}), \qquad a_t \in \mathcal{A}_{\text{gen}} \cup \mathcal{A}_{\text{perc}}, \tag{1}$$

until it commits an answer. The action space includes two classes of actions, both operated through the CLI. $\mathcal{A}_{\text{gen}}$ holds the general coding agent's original actions, such as running Linux commands, writing and executing code, and reading files and images. $\mathcal{A}_{\text{perc}}$ holds the perception toolkit's actions, such as reconstruction, segmentation, and readouts of spatial relations, which the following subsections detail. A perception action returns a full output m and a tool response o^{perc}. The

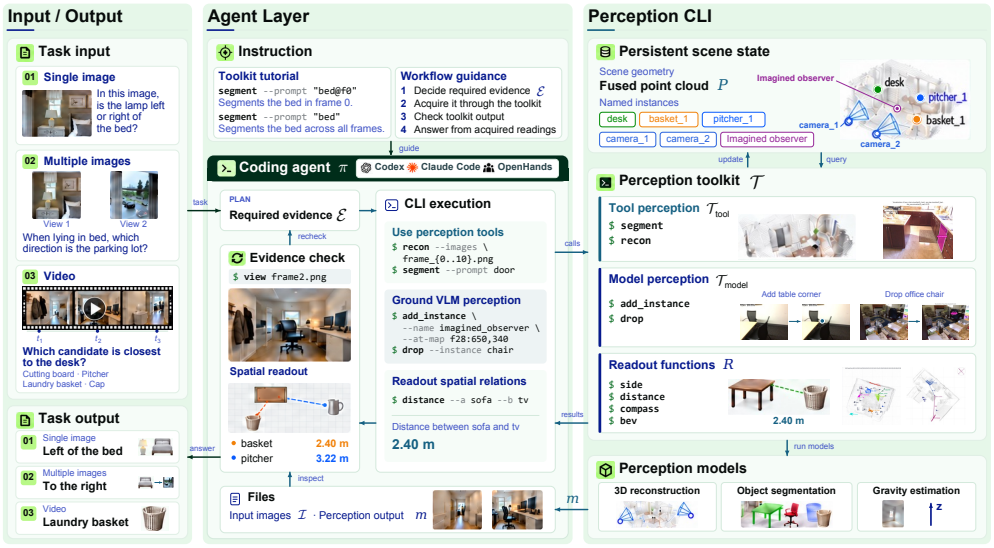


Figure 2: Overview of the framework. The agent keeps a coding agent's full generic freedom over the workspace (viewing images, reading and writing files, writing and running code) and composes perception through the same shell; the toolkit registers everything it perceives into the scene memory and returns interpreted, answer-ready readings.

response o^{perc} joins the context, while the full output m, together with the original frames $\mathcal{I}$, is kept on disk. The agent reads these files through the file-reading actions of $\mathcal{A}_{\text{gen}}$, deciding when and which to read, and what it reads joins the context as o^{gen}. Across these steps the model's own work is the two maps $q \mapsto \mathcal{E}$ and $o^{\text{perc}} \mapsto \mathcal{E}$: deciding what evidence the question requires, and reading each reply into a piece of that evidence. Since this design assumes only the CLI machinery every coding agent provides, our framework works across general coding agent harnesses (§3.3).

2.2 Scene memory kept by the environment

Closing the loop on a multi-view or video question takes many steps, and the evidence gathered early must still be at hand when the answer is committed, so some component has to hold it. The model's context window can serve as that holder, at the price of transcribing every intermediate result into text and carrying it across a growing transcript.

We let the environment do the remembering instead: the toolkit maintains a persistent *scene memory*

$$S = \big(\mathcal{I},\ P,\ \{(\nu_i,\ g_i)\}_i\big), \qquad g_i = \big(\mathbf{p}_i,\ E_i,\ T_i\big), \tag{2}$$

holding the raw frames $\mathcal{I}$, the fused metric point cloud P in a gravity-aligned world frame, and a registry of named instances. Each instance pairs a name ν_i, usable in later calls, with its geometry g_i: an xyz position $\mathbf{p}_i \in \mathbb{R}^3$, an extent E_i (the subset of P belonging to the instance), and a 6-DoF pose $T_i \in \mathrm{SE}(3)$; E_i and T_i may each be left empty. Both the perception toolkit and the model can write to this memory; §2.3 details both routes.

Because the scene memory records each instance's position, extent, and pose, measuring objects and their relations becomes a lookup, computed in closed form from the stored geometry. Readout actions are a family of functions $R_\tau(\nu_1, \ldots, \nu_k;\ S_t)$ with $\tau \in \mathcal{T}$, over one or more named instances, covering properties of a single instance, such as its size and elevation, and relations between instances, such as pairwise distance, relative direction, and motion between frames. A reading is derived fresh from the stored geometry at each call, so the model can complete a readout by remembering only the instance names. Figure 3 lists what one episode's memory holds.

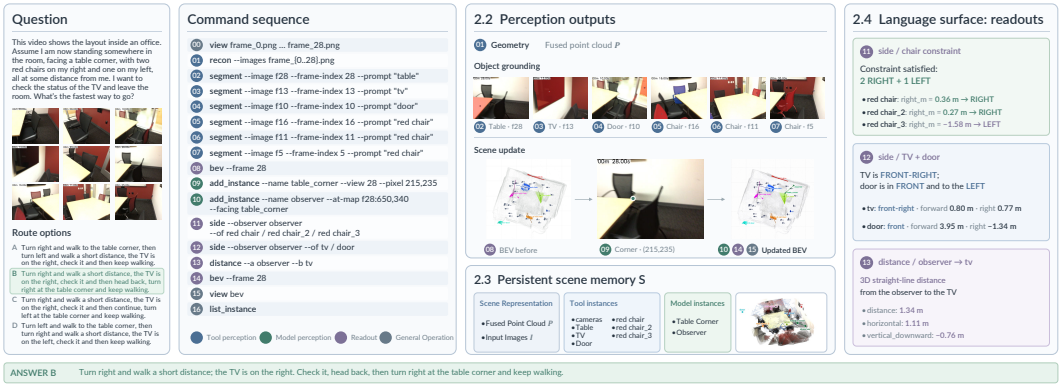


Figure 3: An example question used throughout this section. Left to right: the question, the agent's command sequence colour-coded by who performs the perception, the outputs those calls produce and the scene memory they write, and the readouts that answer the question.

2.3 Dividing perception by competence

The evidence loop involves two kinds of perceptual work with opposite competence profiles. Deciding *what* to measure is semantic work that VLMs do well, such as identifying which objects the question is about and phrasing a segmentation prompt. Producing precise masks and measurements is work that dedicated perception models do better. A principled interface should cut exactly along this line. Formally, a perception action pairs a toolkit command τ with the arguments x the model supplies: the model contributes the map $q \mapsto x$, the grounding step that carries the evidence $\mathcal{E}$ it conceives into terms the toolkit operates on, and the toolkit supplies $(x, S) \mapsto o^{\text{perc}}$. Our interface lets the model write x directly from earlier replies, $o^{\text{perc}} \mapsto x$. The call has semantics

$$(o_t^{\text{perc}},\, S_{t+1}) \;=\; \tau(x_t;\, S_t)\,, \qquad \tau \in \mathcal{T} = \mathcal{T}_{\text{tool}} \cup \mathcal{T}_{\text{model}}, \tag{3}$$

where S_t is the scene memory of §2.2.

We build the division into the interface along these two write routes into the scene memory. *Tool perception* ($\mathcal{T}_{\text{tool}}$) carries the measurement side. The `recon` command fuses all frames into one metric, gravity-aligned scene and registers every camera in the scene memory; the `segment` command grounds objects to pixels, taking the model's own words as prompts, so deciding what to look for remains semantic work. *Model perception* ($\mathcal{T}_{\text{model}}$) enters through the `add-instance` command, which grounds the model's high-level understanding itself. The model names the new instance, such as an object it recognizes or an observer the question imagines, and initializes its position and pose from other named instances or from pixel locations in the images. The toolkit then treats it like any other instance, so a construct that exists only in the model's reading of the question is measured on the same geometric footing as the instances that tool perception grounds. The same route also runs in reverse: with `drop` the model removes an instance the tools registered wrongly, such as a mask on a reflection, so its own look at the tool perception outputs corrects the memory before any reading is taken from it. Figure 3 shows model perception and tool perception apart in one episode. Detailed descriptions of toolkit interfaces are given in Appendix B.

2.4 The language surface: numbers interpreted on arrival

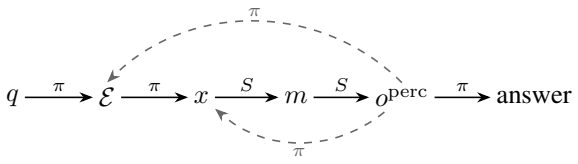
With acquisition routine and memory kept by the environment, the remaining break point is how a reading is consumed. Even when the measurement itself is correct, the model may take it in the wrong reference frame, invert which object it is measured from, or ignore it entirely. A displacement vector alone does not say which object it points from, whether its axes follow gravity or the camera, or which answer option it favors. That knowledge lives outside the numbers, and the side that computed the geometry holds it most reliably. In previous tool-augmented systems (Cho et al., 2026), tool calls return bare values or arrays, leaving the whole step from number to usable spatial statement to the model. In the failures we observed, the model calls the right tool and the tool returns the right geometry, yet the final answer still comes out wrong.

We therefore have every reply interpret its own numbers, handing the model the values and their plain-language reading together. Each command distills its full output m, against the scene memory S, into a compact structured report

$$o^{\text{perc}} \;=\; c_\tau(m;\, S), \tag{4}$$

with a fixed anatomy: a few typed numeric fields with units, together with a *conclusion*, one sentence that restates the measurement in plain spatial language. The conclusion names the entities involved, declares the frame it speaks in (gravity-aligned, camera, or image plane), and carries the reading all the way to the ordinary spatial vocabulary an answer is written in. Figure 3 gives an example: the responses describe relation in words first and keeps the numerical fields that support it, so the model can act on a reading without re-deriving it. And the complete measurements and geometry stay in the full output m on disk, where the model can read them through the same CLI when it needs them.

Together these commitments close the evidence loop. Evidence travels in rounds,



where π is the VLM, S is the scene memory q is the question, $\mathcal{E}$ the evidence the model decides the question requires, x the arguments of a perception call, m the full output the call writes to disk, and o^{perc} the interpreted reply; π marks the model's steps and S the toolkit's, computed against the scene memory. The dashed returns close the loop: the model reads each reply back into its evidence $\mathcal{E}$ and into the arguments of the next call.

3 Experiments

3.1 Setup

Benchmarks. We evaluate S-Agent 2.0 on five benchmarks covering multi-view and video spatial reasoning. For *multi-view* reasoning, **MMSI-Bench** (Yang et al., 2026c) provides multiple images of the same scene and tests whether models integrate evidence across views for positional relationships, geometric attributes, motion, and multi-step reasoning; **MindCube** (Wang et al., 2025b) probes the spatial mental model built from a few sparse views, asking for relations the given views never show directly; **SPAR-Bench** (Zhang et al., 2026a) (used for the harness-swap study) spans a hierarchy of spatial capabilities from basic metric perception to multi-view relational reasoning. For *video* reasoning, **ReVSI** (Zhang et al., 2026b) evaluates 3D spatial reasoning from dynamic observations, emphasizing relations that no isolated frame reliably shows; **MMSI-Video-Bench** (Lin et al., 2025) extends the MMSI setting to long videos, with questions on spatial construction, motion understanding, cross-video memory, and route planning. ReVSI and SPAR-Bench scores follow the official protocols, macro-averaged over question types; the others report overall accuracy.

Models. We use Qwen3.8-27B (FP8, served locally) as the primary backbone; Gemini-3.8-Flash as the stronger backbone for the capability study. The general coding agent harness is OpenHands (Wang et al., 2025c) unless stated otherwise, and the same workspaces and toolkit also run under Codex and Claude Code (§3.3). The perception tools are served by off-the-shelf models, all frozen: Depth Anything 3 (Lin et al., 2026) for the metric reconstruction, π^3_{mos} (Wang et al., 2026; Zhong et al., 2026) for per-pixel motion masks, GeoCalib (Veicht et al., 2024) for the gravity direction, and SAM 3 (Carion et al., 2026) for promptable segmentation.

Baselines. We compare S-Agent 2.0 with three categories of baselines: advanced proprietary VLMs, such as GPT-5.4, Grok 4, and the Gemini series; open-weight general and spatially specialized models, such as the Qwen series and the SenseNova-SI series; and agentic systems, S-Agent (Dai et al., 2026) and SpatialClaw (Cho et al., 2026).

Table 1: MMSI-Bench results by question category (C/O/R denote camera/object/region).

Model	Positional Relationship						Geometric Attr.		Motion		MSR	Avg.
	C-C	O-O	R-R	C-O	O-R	C-R	Meas.	Appr.	Cam.	Obj.		
Proprietary models												
GPT-5.4	41.9	33.0	35.8	49.8	42.4	68.7	54.7	37.4	28.3	40.8	36.4	41.9
GPT-5.6	64.5	42.6	55.6	41.9	44.7	59.0	67.2	40.9	59.5	48.7	57.1	53.2
GPT-6	**73.1**	61.7	**72.8**	58.1	67.1	72.3	**73.4**	**69.7**	**89.2**	65.8	**75.3**	71.0
Grok 4	36.6	35.1	39.5	34.9	45.9	50.6	21.9	22.7	40.5	43.4	38.4	37.8
Gemini 2.5 Pro	38.7	34.0	40.7	44.2	38.8	41.0	62.5	30.3	39.2	25.0	33.3	38.0
Gemini 3 Pro	47.3	48.9	42.0	43.0	37.6	60.2	64.1	39.4	41.9	47.4	37.9	45.2
Gemini 3.8 Flash	68.8	60.6	51.9	51.2	52.9	62.7	64.1	47.0	78.4	61.8	60.1	60.0
Open-weight general models												
Seed 1.6	36.6	36.2	32.1	32.6	42.4	46.9	48.4	33.0	31.1	42.1	40.4	38.5
InternVL3_5-8B	29.0	26.6	29.6	24.4	31.8	25.3	29.7	25.8	14.9	34.2	36.4	29.0
SenseNova-U1-8B-MoT	46.2	41.5	29.6	58.1	38.8	63.9	43.8	21.2	25.7	31.6	26.8	38.0
Qwen3-VL-8B-Instruct	28.0	37.2	32.1	31.4	35.3	38.5	37.5	15.2	27.0	28.9	29.8	31.1
Qwen3-VL-8B-Thinking	31.2	26.6	32.1	29.1	32.9	30.1	50.0	16.7	17.6	23.7	27.3	28.6
Qwen3.5-9B	34.4	36.2	34.6	39.5	38.8	54.2	56.3	28.8	36.5	26.3	28.8	36.5
Qwen3-VL-30B-A3B-Thinking	23.7	31.9	35.8	31.4	36.5	22.9	40.6	19.7	18.9	27.6	31.3	29.4
Qwen3.8-27B	58.1	38.3	44.4	36.0	49.4	55.4	56.2	47.0	67.6	55.3	50.5	50.4
Open-weight spatial models												
SenseNova-SI-1.1-Qwen2.5VL-7B	51.6	29.8	32.1	50.0	29.4	42.2	37.5	28.8	23.0	34.2	18.7	32.8
SenseNova-SI-1.1-Qwen3VL-8B	44.1	38.3	33.3	65.1	38.8	59.0	48.4	24.2	29.7	34.2	22.2	38.1
VST-7B-SFT	39.8	36.2	35.8	37.2	29.4	33.7	29.7	47.0	36.5	35.5	18.2	32.5
Agentic systems												
S-Agent (Qwen3.8-27B)	59.1	47.9	42.0	46.5	34.1	51.8	57.8	47.0	48.6	46.1	49.5	48.3
SpatialClaw (Qwen3.8-27B)	67.7	50.0	54.3	79.1	58.8	79.5	68.8	56.1	77.0	44.7	58.6	62.6
SpatialClaw (Gemini-3.8-Flash)	71.0	68.1	63.0	68.6	64.7	73.5	68.8	50.0	81.1	60.5	66.2	67.0
S-Agent 2.0 (Qwen3.8-27B, ours)	**73.1** (+5.4)	58.5 (+8.5)	58.0 (+3.7)	83.7 (+4.6)	60.0 (+1.2)	74.7 (−4.8)	**73.4** (+4.6)	60.6 (+4.5)	75.7 (−1.3)	52.6 (+7.9)	64.1 (+5.5)	66.5 (+3.9)
S-Agent 2.0 (Gemini-3.8-Flash, ours)	72.0 (+1.0)	**74.5** (+6.4)	67.9 (+4.9)	**89.5** (+20.9)	**68.2** (+3.5)	**81.9** (+8.4)	71.9 (+3.1)	**69.7** (+19.7)	82.4 (+1.3)	**68.4** (+7.9)	71.2 (+5.0)	**74.1** (+7.1)

Table 2: ReVSI results by question type. Numerical types (object counting, absolute distance, object size, room size) use the official mean relative accuracy; the remaining types are multiple choice.

Model	Frames	Numerical Question				Multiple-Choice Question			Avg.
		Obj. Cnt.	Abs. Dist.	Obj. Size	Room Size	Rel. Dist.	Rel. Dir.	Route Plan	
Proprietary models									
GPT-5.2	64	56.2	41.5	73.9	63.0	48.4	34.9	38.2	50.9
GPT-5.4	64	70.3	40.2	73.3	59.8	57.8	43.5	33.3	54.0
GPT-5.6	64	80.6	64.4	80.3	71.7	71.8	71.0	63.9	72.0
Gemini 3 Flash	1 FPS	65.7	53.1	77.6	52.8	64.6	47.9	41.8	57.6
Gemini 3 Pro	1 FPS	60.1	54.7	79.3	51.9	68.1	56.0	56.4	60.9
Gemini 3.8 Flash	64	79.7	70.1	83.0	68.9	81.6	78.3	77.8	77.0
GPT-6	64	**89.2**	72.5	83.4	75.0	87.4	95.7	86.1	84.2
Open-weight general models									
Qwen3-VL-8B-Instruct	64	40.4	52.3	69.0	45.1	57.1	39.5	40.5	49.1
Qwen3-VL-32B-Instruct	64	46.9	65.0	70.4	55.8	53.8	34.0	47.3	53.3
InternVL3.5-8B	64	43.3	54.6	64.2	47.6	45.0	36.3	44.4	47.9
InternVL3.5-38B	64	43.8	60.6	70.2	58.4	57.4	45.9	42.7	54.1
LLaVA-Video-7B-Qwen2	64	31.3	1.4	52.5	16.7	38.3	33.3	38.4	30.3
LLaVA-Video-72B-Qwen2	64	40.1	29.6	59.3	27.9	39.6	24.8	43.0	37.8
Qwen3.8-27B	64	63.5	52.1	71.1	53.5	64.1	63.8	61.1	61.3
Spatially specialized models and base models									
Cambrian-S-7B	128	48.4	60.5	65.5	46.7	37.1	48.5	37.0	49.1
Qwen2.5-VL-7B-Instruct	4 FPS	36.9	15.0	49.7	29.0	31.5	29.5	36.7	32.6
VST-7B-SFT	4 FPS	35.4	52.6	67.9	47.2	49.2	36.9	35.4	46.4
Qwen2.5-VL-7B-Instruct	32	34.3	21.7	45.5	35.1	32.6	33.7	34.1	33.9
SpaceR-7B (SG-RLVR)	32	30.7	34.5	52.0	18.6	22.8	34.5	20.2	30.5
Qwen2.5-VL-3B-Instruct	16	18.7	15.6	16.8	–	33.2	34.3	–	23.7
Spatial-MLLM-4B-135k	16	40.7	45.3	46.8	–	32.3	37.4	–	40.5
Spatial-MLLM-4B-820k	16	41.5	40.0	53.1	–	30.7	39.2	–	40.9
LLaVA-Video-7B-Qwen2	32	29.9	1.5	53.0	19.3	39.1	33.8	38.8	30.8
VLM3R-7B	32	41.6	61.6	64.8	52.5	46.5	49.5	34.1	50.1
Agentic systems									
S-Agent (Qwen3.8-27B)	64	63.0	58.1	60.9	57.4	63.6	69.1	44.4	59.5
SpatialClaw (Qwen3.8-27B)	64	74.8	79.5	73.4	59.1	84.9	77.8	72.2	74.5
SpatialClaw (Gemini-3.8-Flash)	64	80.3	**89.9**	**86.6**	75.2	87.9	63.3	75.0	79.7
S-Agent 2.0 (Qwen3.8-27B, ours)	64	75.2 (+0.4)	76.7 (−2.8)	78.7 (+5.3)	67.8 (+8.7)	88.3 (+3.4)	95.2 (+17.4)	94.4 (+22.2)	82.3 (+7.8)
S-Agent 2.0 (Gemini-3.8-Flash, ours)	64	86.8 (+6.5)	88.2 (−1.7)	86.1 (−0.5)	**82.2** (+7.0)	**95.1** (+7.2)	**97.6** (+34.3)	**97.2** (+22.2)	**90.5** (+10.8)

3.2 MAIN RESULT

Table 1 shows the accuracy on MMSI-Bench, overall and by question type. With the 27B open backbone and no training S-Agent 2.0 is ahead of every agentic baseline and of every proprietary model other than GPT-6; with Gemini-3.8-Flash it has the highest average in the table, while GPT-6 keeps the lead on region-to-region relations, camera motion and multi-step reasoning. The parentheses give the change against SpatialClaw on the same backbone.

Table 3: Comparison with agentic spatial reasoning baselines, Qwen3.8-27B backbone.

Method	MMSI	MindCube	SPAR	MMSI-Vid	ReVSI
pySpatial	36.3	66.3	42.1	26.9	39.3
SpaceTools	43.2	58.6	55.3	39.3	64.4
S-Agent	48.3	75.5	63.0	37.9	59.5
SpatialClaw	62.6	83.2	70.2	53.1	74.5
S-Agent 2.0	**66.5** (+3.9)	**88.3** (+5.1)	**72.4** (+2.2)	**56.4** (+3.3)	**82.3** (+7.8)

Table 4: S-Agent 2.0 under different coding-agent harnesses, Qwen3.8-27B backbone.

Benchmark	OpenHands	Codex	Claude Code
MMSI-Bench	66.5	67.0	63.8
SPAR-Bench	72.4	71.8	72.8
ReVSI	82.3	76.6	79.1

Table 5: Tools against no tools on ReVSI and MMSI-Bench, for two backbones and two harnesses.

		Qwen3.8-27B		Gemini-3.8-Flash	
Harness	Tools	ReVSI	MMSI	ReVSI	MMSI
SpatialClaw	no	61.3	50.4	77.0	60.0
	with	74.5	62.6	79.7	67.0
general coding agent	no	61.8	52.6	80.9	62.9
	with	**82.3**	**66.5**	**90.5**	**74.1**

Table 6: Ablations on interface design. Qwen3.8-27B backbone.

Variant	MMSI	ReVSI
S-Agent 2.0	**66.5**	**82.3**
no scene memory	63.0	79.8
no interpretation readout	65.2	81.1
no model perception interface	57.7	76.4

Table 2 shows the accuracy on ReVSI, overall and by question type. S-Agent 2.0 is ahead of every agentic baseline on both backbones, and with Gemini-3.8-Flash it has the highest average in the table. The margins over the specialized baseline are largest on the types that need cross-object geometry, relative direction and route planning; the one deficit is absolute distance, which is also where GPT-6 falls furthest behind.

Table 3 shows the accuracy on all five benchmarks against the agentic baselines, where S-Agent 2.0 is ahead on every one.

3.3 Ablations

Different general coding agent harness. We rerun the benchmarks with the whole harness replaced by another production coding agent, Codex or Claude Code, each with its own agent loop, prompting and context management, driving the identical workspaces and toolkit. Table 4 shows S-Agent 2.0 works under every one.

Tools against no tools. Table 5 varies the toolkit against the same agent on the same items. The toolkit is worth a large gain on both benchmarks on Qwen3.8-27B, and our score with tools keeps rising as the backbone grows more capable. SpatialClaw scores below ours on every backbone and benchmark, and its tool gain shrinks far more as the backbone grows.

Interface components. Table 6 removes one commitment of §2 at a time and leaves the rest of the system untouched. Without the scene memory the agent carries every reading through its own context; without the interpreted readout the replies come back as bare values; without the model perception interface the model cannot add instances to the scene or drop them from it. Every removal costs accuracy, and the ranking matches the diagnosis of §1: on the surface only interpretation goes wrong, but underneath the acquired evidence is never fully used. Removing the interpreted readout costs the least (-1.3 and -1.2); removing the scene memory costs more (-3.5 and -2.5), since readings carried through the context are lost before the answer is written; removing the model perception interface costs the most (-8.8 and -5.9). Closing the loop needs both the interpretation design and the scene memory.

3.4 Qualitative analysis

Qualitative comparison with SpatialClaw. On the same backbone and the same perception backends, the baseline acquires the evidence and still does not close the loop (Figure 4, left): the tools return the right masks and the right positions, and the step from those numbers to the asked relation is left to the model, which states a right-hand rule correctly and implements it in reverse, or reaches the correct direction and lets a second look at the pictures overturn it. Our runs close the same loop because that step is not left to the model: the question's premise enters the scene as an oriented in-



Figure 4: Qualitative analysis. Left: two items where SpatialClaw calls the right tools and gets the right perception, and still answers wrong, with S-Agent 2.0 correct on both (gemini-3.8-flash on both sides). Top right: S-Agent 2.0 cross-checking its own perception against the scene memory, dropping the wrong masks before grounding the observer and reading the answer. Bottom right: the two remaining grounding failure classes, a wrongly imagined viewpoint and a quantity read in the wrong sense. Zoom in for better readability.

stance or a compass anchor, and the relation comes back in the words the options use. Seven further items, with the command-line record and the intermediate images of both agents, are in Appendix E.

Cross-checking the two kinds of perception. The memory closes the loop only if what it holds is right, and the agent checks that itself. It compares what it sees in a perception output against what the registry holds and edits the memory when the two disagree (Figure 4, top right): it drops instances the output shows to be wrong, and keeps a suspected duplicate once the output shows a genuine second object. The habit is the coding agent's inspect-and-revise loop applied to the scene, and it is what joins the model's perception with the tools' before any reading is taken.

What is left is grounding. With acquisition, memory and interpretation carried by the interface, the errors that remain sit where the question first enters the loop (Takeaway 3, Figure 4, bottom right). The agent fails to simulate an imagined viewpoint the question requires, registering an observer with a borrowed heading, or it measures the right quantity in the wrong sense, reading an object's elevation above the floor where the question asks for the object's own height.

4 Conclusion

We built a spatial reasoning agent from a stock general coding agent and a command-line perception toolkit that closes the evidence loop for the model, joining the tools' measurements with the model's own perception, keeping every reading in a persistent scene memory, and returning each reading interpreted in spatial language. Experiments show that with a 27B open backbone and no training the agent is ahead of every agentic baseline on all five benchmarks, and that with Gemini-3.8-Flash it has the highest MMSI-Bench and ReVSI scores of any model we compare against, including proprietary models such as GPT-6 and GPT-5.6. Its gain from the toolkit remains substantial on the stronger backbone, where the specialized baseline's shrinks far more, and the same toolkit works under different coding agent harnesses.

AI USE STATEMENT

In this work, we have not used generative AI tools to implement methods, to generate synthetic data sets, to develop theoretical models or conceptual frameworks, to propose or refine hypotheses, to design or provide feedback on research methodology or experiments, to clean or reformat data sets, to support qualitative or thematic data analysis, or to interpret results; formulating mathematical claims, providing ingredients for or assisting in the writing of proofs, and assisting with translation are not applicable to this work. Additionally, we used generative AI tools to create or edit software code, to edit the paper to improve readability, and to create or modify scientific figures. We have reviewed all AI-assisted work: AI-generated code was read, verified and tested by the authors before use, every passage of AI-edited text was checked by the authors against the underlying experiments, and AI-assisted figures were checked against the data they display. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Code and the trajectories of all reported runs will be released upon acceptance. Appendix B documents the toolkit's command-line interface.

REFERENCES

Anthropic. Claude 3.7 sonnet and claude code. https://www.anthropic.com/news/claude-3-7-sonnet, February 2025. Accessed: 2026-09-23.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023. URL https://arxiv.org/abs/2307.15818.

Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 11618–11628. IEEE, 2020.

Zhongang Cai, Ruisi Wang, Chenyang Gu, Fanyi Pu, Junxiang Xu, Yubo Wang, Wanqi Yin, Zhitao Yang, Chen Wei, Qingping Sun, Tongxi Zhou, Jiaqi Li, Hui En Pang, Oscar Qian, Yukun Wei, Zhiqian Lin, Xuanke Shi, Kewang Deng, Xiaoyang Han, Zukai Chen, Xiangyu Fan, Hanming Deng, Lewei Lu, Liang Pan, Bo Li, Ziwei Liu, Quan Wang, Dahua Lin, and Lei Yang. Scaling spatial intelligence with multimodal foundation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris Coll-Vinent, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, RISHI HAZRA, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollar, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. Sam 3: Segment anything with concepts. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 138846–138923, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/e0982cbc81401df3430ee1ff780dc7a2-Paper-Conference.pdf.

Boyuan Chen, Zhuo Xu, Sean Kirmani, Brain Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14455–14465, 2024.

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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 37109–37120, 2026a.

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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 38689–38699, 2026b.

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. *Advances in Neural Information Processing Systems*, 37:135062–135093, 2024.

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.

Yalun Dai, Hao Li, Shulin Tian, Runmao Yao, Yuhao Dong, Fangzhou Hong, Zhaoxi Chen, Fangfu Liu, Baoliang Tian, Dingwen Zhang, Tao Wang, Kim-Hui Yap, and Ziwei Liu. S-agent: Spatial tool-use elicits reasoning for spatial intelligence. *CoRR*, abs/2606.20515, 2026. doi: 10.48550/ARXIV.2606.20515. URL https://doi.org/10.48550/arXiv.2606.20515.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language model, 2023. URL https://arxiv.org/abs/2303.03378.

Mengfei Du, Binhao Wu, Zejun Li, Xuan-Jing Huang, and Zhongyu Wei. Embspatial-bench: Benchmarking spatial understanding for embodied tasks with large vision-language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 346–355, 2024.

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 *European Conference on Computer Vision*, pp. 148–166. Springer, 2024.

Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 3354–3361. IEEE, 2012.

Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Nan Duan, and Weizhu Chen. Critic: Large language models can self-correct with tool-interactive critiquing. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 57734–57811, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/fef126561bbf9d4467dbb8d27334b8fe-Paper-Conference.pdf.

Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14953–14962. IEEE, 2023.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.

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. In *International Conference on Learning Representations*, volume 2026, pp. 76566–76592, 2026.

Haotong Lin, Sili Chen, Jun Hao Liew, Donny Y. Chen, Zhenyu Li, Yang Zhao, Sida Peng, Hengkai Guo, Xiaowei Zhou, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=yirunib8l8.

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.

Pan Lu, Bowen Chen, Sheng Liu, Rahul Thapa, Joseph Boen, and James Zou. Octotools: An agentic framework with extensible tools for complex reasoning. *arXiv preprint arXiv:2502.11271*, 2025.

Zhanpeng Luo, Ce Zhang, Silong Yong, Cunxi Dai, Qianwei Wang, Haoxi Ran, Guanya Shi, Katia 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 *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19446–19455. IEEE, 2025.

OpenAI. Introducing codex. https://openai.com/index/introducing-codex/, May 2025. Accessed: 2026-09-23.

Kun Ouyang, Yuanxin Liu, Haoning Wu, Yi Liu, Hao Zhou, Jie Zhou, Fandong Meng, and Xu Sun. Spacer: Reinforcing mllms in video spatial reasoning. *arXiv preprint arXiv:2504.01805*, 2025.

Fernando Ropero, Erkin Turkoz, Daniel Matos, Junqing Du, Antonio Ruiz, Yanfeng Zhang, Lu Liu, Mingwei Sun, and Yongliang Wang. Riemind: Geometry-grounded spatial agent for scene understanding. *arXiv preprint arXiv:2603.15386*, 2026.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180, 2023.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

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 *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15768–15780. IEEE, 2025.

Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11854–11864. IEEE, 2023.

Alexander Veicht, Paul-Edouard Sarlin, Philipp Lindenberger, and Marc Pollefeys. Geocalib: Learning single-image calibration with geometric optimization. In *European Conference on Computer Vision*, pp. 1–20. Springer, 2024.

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, 2025a.

Qineng Wang, Baiqiao Yin, Pingyue Zhang, Jianshu Zhang, Kangrui Wang, Zihan Wang, Jieyu Zhang, Keshigeyan Chandrasegaran, Han Liu, Ranjay Krishna, Saining Xie, jiajun Wu, Li Fei-Fei, and Manling Li. Spatial mental modeling from limited views. In *The Fourteenth International Conference on Learning Representations*, 2025b.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better llm agents. In *International Conference on Machine Learning (ICML)*, 2024.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Daniel Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. Openhands: An open platform for ai software developers as generalist agents. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 65882–65919, 2025c. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/a4b6ad6b48850c0c331d1259fc66a69c-Paper-Conference.pdf.

Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. π^3: Permutation-equivariant visual geometry learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=DTQIjngDta.

Diankun Wu, Fangfu Liu, Yi-Hsin Hung, and Yueqi Duan. Spatial-mllm: Boosting mllm capabilities in visual-based spatial intelligence. *Advances in neural information processing systems*, 38: 13569–13597, 2026a.

Junfei Wu, Jian Guan, Kaituo Feng, Qiang Liu, Shu Wu, Liang Wang, Wei Wu, and Tieniu Tan. Reinforcing spatial reasoning in vision-language models with interwoven thinking and visual drawing. *Advances in Neural Information Processing Systems*, 38:143297–143330, 2026b.

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, 2025.

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

Rui Yang, Ziyu Zhu, Yanwei Li, Jingjia Huang, Shen Yan, Siyuan Zhou, Zhe Liu, Xiangtai Li, Shuangye Li, Wenqian Wang, Yi Lin, and Hengshuang Zhao. Visual spatial tuning. In *European Conference on Computer Vision*, pp. 192–211. Springer, 2026a.

Shusheng Yang, Jihan Yang, Pinzhi Huang, Ellis L Brown II, Zihao Yang, Yue Yu, Shengbang Tong, Zihan Zheng, Yifan Xu, Muhan Wang, Rob Fergus, Yann LeCun, Li Fei-Fei, and Saining Xie. Cambrian-s: Towards spatial supersensing in video. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=rBFDvZu6pb.

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. In *International Conference on Learning Representations*, volume 2026, pp. 157051–157088, 2026c.

Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. Mm-react: Prompting chatgpt for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*, 2023.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Jinhui Ye, Zihan Wang, Haosen Sun, Keshigeyan Chandrasegaran, Zane Durante, Cristobal Eyzaguirre, Yonatan Bisk, Juan Carlos Niebles, Ehsan Adeli, Li Fei-Fei, et al. Re-thinking temporal search for long-form video understanding. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8579–8591. IEEE, 2025.

Hang Zhang, Xin Li, and Lidong Bing. Video-llama: An instruction-tuned audio-visual language model for video understanding. In *Proceedings of the 2023 conference on empirical methods in natural language processing: system demonstrations*, pp. 543–553, 2023.

Jiahui Zhang, Yurui Chen, Yueming Xu, Ze Huang, Jilin Mei, Chunhui 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. *Advances in Neural Information Processing Systems*, 38, 2026a.

Peiyuan Zhang, Kaichen Zhang, Bo Li, Guangtao Zeng, Jingkang Yang, Yuanhan Zhang, Ziyue Wang, Haoran Tan, Chunyuan Li, and Ziwei Liu. Long context transfer from language to vision. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=30RAWQVGlx.

Yiming Zhang, Jiacheng Chen, Jiaqi Tan, Yongsen Mao, Wenhu Chen, and Angel X Chang. Revsi: Rebuilding visual spatial intelligence evaluation for accurate assessment of vlm 3d reasoning. In *Forty-third International Conference on Machine Learning*, 2026b.

Zaibin Zhang, Yuhan Wu, Lianjie Jia, Yifan Wang, Zhongbo Zhang, Yijiang Li, Binghao Ran, Fuxi Zhang, Zhuohan Sun, Yizhuang Peng, Zhenfei Yin, Lijun Wang, and Huchuan Lu. Think3d: Thinking with space for spatial reasoning, 2026c. URL https://arxiv.org/abs/2601.13029.

Shitian Zhao, Haoquan Zhang, Shaoheng Lin, Ming Li, Qilong Wu, Kaipeng Zhang, and Chen Wei. Pyvision: Agentic vision with dynamic tooling. *arXiv preprint arXiv:2507.07998*, 2025.

Xingguang Zhong, Liren Jin, Marija Popovic, Jens Behley, and Cyrill Stachniss. Dynamic visual slam using a general 3d prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21805–21815, 2026.

A Related Work

Spatial reasoning in vision-language models. Spatial competence remains one of the weaker abilities of vision-language models (Chen et al., 2024; Cheng et al., 2024; Song et al., 2025), and most of the response has been to put the capability into the weights. One line distills 3D annotations into spatial instruction data at scale, as in SpatialVLM (Chen et al., 2024), SpatialRGPT (Cheng et al., 2024), Cambrian-S (Yang et al., 2026b), and SenseNova-SI (Cai et al., 2026). A second line reshapes the backbone itself with geometry-aware modelling or spatially targeted tuning, as in Spatial-MLLM (Wu et al., 2026a) and VST (Yang et al., 2026a). A third line shapes the training signal through reinforcement learning, verifiable rewards, or curricula, as in SpaceR (Ouyang et al., 2025), ViLaSR (Wu et al., 2026b), MindCube (Wang et al., 2025b), and SpatialLadder (Li et al., 2026). These models answer in a single forward pass out of what training left in their parameters, taking no measurement of the scene in front of them, so a new perception backend or a shifted task distribution asks for another round of training. S-Agent 2.0 leaves the backbone untouched and moves the work to inference time, where the agent acquires scene evidence on demand and every reading it acts on stays grounded in the tools' geometry.

Tool-augmented visual agents and code as action. Tool-use agents interleave reasoning with calls to external modules, following the template that ReAct (Yao et al., 2023) set on the language side. On the visual side, VisProg (Gupta & Kembhavi, 2023) and ViperGPT (Surís et al., 2023) synthesize a whole program in one pass and then execute it, while MM-ReAct (Yang et al., 2023), HuggingGPT (Shen et al., 2023), and OctoTools (Lu et al., 2025) dispatch requests turn by turn through a structured tool menu, and PyVision (Zhao et al., 2025) writes its tooling as it goes. CodeAct (Wang et al., 2024) compared action formats directly and found executable code to work better than JSON or free text for general-purpose agents, which is the format a coding agent uses natively. The tools all of these systems reach for target 2D recognition, so their controllers are never asked to hold a metric 3D scene across views and time, which is what a spatial question demands.

Agentic spatial reasoning. A recent line brings tool use to spatial questions. VADAR (Marsili et al., 2025) synthesizes a task-specific Pythonic API and then a program over it, and pySpatial (Luo et al., 2026) composes reconstruction, camera-pose recovery, and novel-view synthesis into a 3D visual program. GCA (Chen et al., 2026b) splits the model into a part that formalizes the question as geometric constraints and a part that executes tool calls within them, RieMind (Ropero et al., 2026) queries an explicit 3D scene graph through typed geometric operations, and SpaceTools (Chen et al., 2026a) fine-tunes a VLM to coordinate a fixed tool set through supervised demonstrations followed by interactive reinforcement learning. Think3D (Zhang et al., 2026c) selects camera viewpoints, renders a reconstructed point cloud from them, and reasons over the rendered images. Closest to us, S-Agent (Dai et al., 2026) schedules 3D operations across the views and time steps of multi-view sets and long videos, and SpatialClaw (Cho et al., 2026) drives the VLM through a hand-designed plan, act, and reflect loop over a persistent code kernel stocked with 3D perception and geometry utilities. What these systems share is a controller written for the purpose and a design effort spent on acquisition, on ever richer perception and geometry libraries, while remembering and interpreting the evidence is left to the model and the model's own perception stays in a separate channel. S-Agent 2.0 writes no controller and spends its design effort on closing the evidence loop at the interface: the model's own perception enters the same scene as the tools' measurements, every reading persists in a scene memory that later calls query by name, and each reading comes back already interpreted in spatial language.

General-purpose coding agents. Software-engineering agents such as SWE-agent (Yang et al., 2024) and OpenHands (Wang et al., 2025c), developed against benchmarks such as SWE-bench (Jimenez et al., 2024), have matured into harnesses that supply a shell, a file system, code execution, long-horizon context management, and a tool-calling loop that the backbone models are trained to follow. S-Agent 2.0 adopts such a harness unchanged and adds perception as ordinary command-line tools, which is what lets the same method run under more than one of them (§3.3). Two habits of these agents carry over to spatial reasoning: viewing an image is an ordinary step in the same stream as the tool replies, and inspecting an output and revising it is a reflex; together they are what joins the model's own perception with the tools' before a reading is taken (§3.4).

Long-video and multi-view perception. Two families of methods supply the visual evidence these questions need. Frame-compression methods sample, retrieve, or summarize a limited set of frames for a long-context VLM, as in Video-LLaMA (Zhang et al., 2023), LongVA (Zhang et al., 2025), and T* (Ye et al., 2025), which keeps the context tractable at the risk of dropping the frames a question turns on. Reconstruction-first methods build an explicit 3D representation with multi-view geometry or feed-forward models such as VGGT (Wang et al., 2025a) and Depth Anything 3 (Lin et al., 2026), which grounds the geometry well while paying the reconstruction cost on every question. Both families hand their output to the model as fixed context, leaving association across views and metric comparison to implicit reasoning. S-Agent 2.0 uses reconstruction and promptable segmentation backends of this kind as tool perception (§2.3), with the agent deciding per question which of them to call and what to ground; what they measure and what the model registers itself land in one scene memory, and every reading comes back interpreted.

Spatial reasoning benchmarks. Evaluation has moved from single-image probes such as BLINK (Fu et al., 2024) and EmbSpatial (Du et al., 2024) to settings where the evidence is spread over many observations. MMSI-Bench (Yang et al., 2026c) asks for evidence integrated across images of one scene, MindCube (Wang et al., 2025b) probes the mental model built from a few sparse views, and SPAR-Bench (Zhang et al., 2026a) spans a hierarchy from metric perception to multi-view relational reasoning. For video, VSI-Bench (Yang et al., 2025) opened the setting, ReVSI (Zhang et al., 2026b) rebuilds its evaluation for a more accurate assessment of 3D reasoning, and MMSI-Video-Bench (Lin et al., 2025) extends the MMSI setting to long videos with questions on spatial construction, motion, cross-video memory, and route planning. We evaluate on five of these (§3.1).

B TOOLKIT INTERFACE

The toolkit is one executable, `python3 /app/perception.py <command> [flags]`, run from the agent's shell like any other program. Every command reads and writes the scene memory S of §2.2; its reply is a JSON document on standard output, and every image it renders (segmentation overlays, top-down maps) is written to disk for the agent to open with its ordinary file viewer. The instruction u is a tutorial over these commands; this appendix lists the interface itself. Table 7 gives every command the agent is taught, grouped as in §2.3.

Instance names. Every argument that refers to something in the scene is a name. `recon` registers `camera:f<N>` for the camera of frame N (position and orientation) and `scene` for the room centre. `segment` registers each prompt's label as an instance (`bed`, `desk`); a second, separate object under the same label becomes `<label>_2`, and `f<N>:<label-or-id>` names one frame's view of an object, by label or by the id stamped on that frame's overlay. `add_instance` names are chosen by the model. Two forms exist for `compass` only: `camplane:f<N>`, the image plane of camera N, and `<name>|plane`, the plane fitted through an instance's points.

Positions and orientations for `add_instance`. A position comes from one of three sources: `--at-map f<N>:PX,PY`, a pixel of the top-down map rendered from frame N (height filled in from the reconstruction at that spot); `--view N --pixel X,Y`, a pixel of the original frame lifted to its 3D point; or `--at-camera N`, the position of camera N. An orientation is a primary axis taken exactly and a secondary direction adjusted to sit orthogonal to it; the axes are the instance's own (forward/back, right/left, top/bottom) and a direction value is any existing instance, the literal `gravity`, or `bev:f<N>:START:END`, an arrow between two points on the map from frame N. `--facing V` is the upright shorthand (gravity fills the vertical, `V` gives the heading), `--primary AXIS=V --secondary AXIS=V` the general form. A second `add_instance` call with the same `--name` and only orientation flags orients the existing instance in place.

Reply anatomy. Every reply has the same shape: a status flag, typed numeric fields in metres on the reconstruction's own scale (a `units` field says whether the scale has been calibrated with `--known`), and a `conclusion` sentence per target that restates the measurement in spatial language. The `side` reply of the episode in Figure 4 (bottom right) reads, verbatim:

```
$ python3 /app/perception.py side --observer sitting_person --of desk
{
  "ok": true,
```

Table 7: The toolkit's command-line interface. Flags marked $^+$ are repeatable. $P, \mathcal{I}$ and the instance registry are the components of the scene memory S (Eq. 2).

Command	Arguments	Effect on S and reply
Tool perception ($\mathcal{T}_{\text{tool}}$)		
`recon`	none (all frames under `/app`)	Runs Depth Anything 3 (Lin et al., 2026) on all frames for metric depth and camera poses, GeoCalib (Veicht et al., 2024) for the gravity direction and π^3_{mos} (Wang et al., 2026; Zhong et al., 2026) for per-pixel motion masks; fuses them into the metric, gravity-aligned point cloud P and registers `camera:f<N>` with its pose and `scene`. Returns the instance list, with an `overlap_warning` naming the seam when two views share almost no covisible surface.
`segment`	`--prompt TEXT[@f<N>]`$^+$, `--max-masks K`	Runs SAM 3 (Carion et al., 2026) with each text prompt on the named frame or on all frames, lifts every mask onto P and registers each label as an instance with extent E_i (same-label regions within 0.75 m merge), tagged `static` or `moving` from the motion masks. Returns, per frame, the overlay PNG path and the region ids with label and centroid, then the updated registry.
Model perception ($\mathcal{T}_{\text{model}}$)		
`add_instance`	`--name`, `--source`; position `--at-map f<N>:PX,PY` | `--view N --pixel X,Y` | `--at-camera N`; orientation `--facing V` | `--primary A=V --secondary A=V`	Registers an instance at the chosen $\mathbf{p}_i$, with pose T_i when an orientation is given; called again with the same name and only orientation flags, re-orients it in place. Returns the updated registry.
`drop`	`--instance f<N>:<id>`$^+$	Removes those instances permanently. Returns the registry after the drop.
Readouts R_τ (closed form over the stored geometry)		
`list_instance`	none	No change. Returns each instance's name, source, and whether it is oriented.
`bev`	`--frame N`, `--erase f<M>:X`$^+$	No change. Returns a top-down map PNG in camera N's ground frame: static points of all frames plus what frame N saw, cameras as wedges, instances as labelled dots, added instances as diamonds with their facing arrow, and the compass rose once anchored.
`distance`	`--a X --b Y`, `--known D`	`--known` stores the stated distance as the scale calibration. Returns straight-line, horizontal and vertical metres and a conclusion.
`depth`	`--view N` | `--observer X`; `--of Y`$^+$; `--known D`	`--known` calibrates scale. Returns each target's signed depth along the observer's forward axis, a nearest-to-farthest `order`, and a conclusion.
`side`	`--view N` | `--observer X`; `--of Y`$^+$	No change. Returns, per target, `forward_m`, `right_m`, `up_m` in the observer's frame (a camera's is its gravity-aligned ground heading), the three axis verdicts, the eight-way `direction`, and a conclusion.
`elevation`	`--of REF --of Y`$^+$	No change. Returns each target's signed height above the reference along gravity, a lowest-to-highest `order`, and a conclusion.
`motion`	`--of A --of B`	No change. For two cameras, returns the translation in the first camera's ground frame, the turn and tilt, and the path of the raw extrinsics; for one object in two frames, its shift as seen from the first camera, its `rotation` about the vertical and `image_rotation` in the image plane; each with a conclusion.
`compass`	`--start A --end B --orientation DIR`, `--query "A->B"`$^+$, `--start-plane`, `--end-plane`	The anchor persists the compass mapping for later calls and maps. Returns the `direction` of each query, the eight compass words ranked by fit (`nearest`), and a conclusion that, for an object-to-camera-plane query, states which side of the object that view sees.

Table 8: Toolkit usage per episode. Calls per episode and, in parentheses, the share of episodes that issue the command at least once. Image views count the agent's own `view` calls on benchmark frames, segmentation overlays and top-down maps. MMSI-Bench and ReVSI, 1000 items each.

	Qwen3.8-27B, MMSI	Gemini-3.8-Flash, MMSI	Gemini-3.8-Flash, ReVSI
`recon`	1.00 (100%)	1.00 (100%)	1.00 (100%)
`segment`	1.75 (93%)	1.47 (92%)	2.25 (100%)
`add_instance`	1.47 (36%)	0.72 (26%)	1.48 (48%)
`drop`	0.37 (26%)	0.27 (21%)	0.53 (40%)
`list_instance`	0.29 (24%)	0.52 (45%)	1.11 (91%)
`bev`	1.66 (94%)	1.67 (94%)	1.74 (98%)
`distance`	1.53 (46%)	0.37 (17%)	2.86 (75%)
`depth`	0.18 (12%)	0.02 (2%)	0.01 (1%)
`side`	1.66 (62%)	1.06 (53%)	0.50 (29%)
`elevation`	0.17 (7%)	0.11 (7%)	0.11 (8%)
`motion`	1.08 (73%)	0.91 (66%)	0.06 (3%)
`compass`	0.55 (25%)	0.69 (25%)	0.01 (0%)
frame views	2.23	2.52	11.43
overlay views	2.12	1.26	2.70
map views	1.59	1.40	1.36


```
  "observer": "sitting_person",
  "sides": [
    {
      "target": "desk",
      "forward_m": 1.04, "right_m": -1.03, "up_m": 0.21,
      "front_or_behind": "front", "left_or_right": "left",
      "above_or_below": "above", "direction": "front-left",
      "conclusion": "desk is 1.04 m in FRONT of sitting_person, 1.03 m
        to its LEFT and 0.21 m ABOVE it -- overall to its FRONT-LEFT"
    }
  ],
  "units": "metres on the reconstruction's own metric scale
    (uncalibrated)"
}
```


The executable also exposes lower-level commands (`detect`, `measure`, `match`, `egomotion`, `gravity`, `render`) that the instruction tells the agent not to use.

C TOOLKIT USAGE

Table 8 counts, over every episode of three full-benchmark runs of S-Agent 2.0 under OpenHands, how often the agent issues each command of Table 7 and how often it opens each kind of image. Building the scene is universal: `recon`, `segment` and `bev` appear in over ninety percent of episodes on every run. The readouts follow the questions: MMSI-Bench, whose items ask about relations between views and viewpoints, draws `side`, `motion` and `compass`; ReVSI, whose items ask about distances, sizes and counts in a video, draws `distance` and `list_instance` instead. The model perception interface is used in a quarter to a half of the episodes, and `drop` in a fifth to two fifths, so registry edits are a routine part of an episode rather than an exception. The lower-level commands the instruction reserves are called in under two percent of all calls.

D SPATIALCLAW WITH OUR READOUTS

To separate what the readouts contribute from what the rest of the interface contributes, we add our readout commands to SpatialClaw's kernel as plain functions and leave its harness, prompts and perception backends untouched. The transplanted set is `distance`, `depth`, `side`, `elevation`, `motion` and `compass`; each returns the same interpreted reply as in Table 7, packed into a Python

Table 9: SpatialClaw with our readouts transplanted into its kernel, against SpatialClaw and S-Agent 2.0. Qwen3.8-27B backbone.

System	ReVSI	MMSI
SpatialClaw	74.5	62.6
SpatialClaw + our readouts	75.5	62.9
S-Agent 2.0	**82.3**	**66.5**

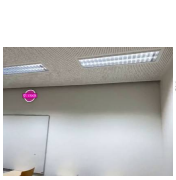
dictionary with the same keys (the numeric fields, the per-target `conclusion` and `units`), so the model reads it in its kernel exactly as our agent reads the JSON on standard output. `list_instance` is excluded because it reads the scene memory that SpatialClaw does not keep, and `bev` because SpatialClaw already renders a top-down map of its own. Table 9 compares the three systems on the same items as Table 5, all on the Qwen3.8-27B backbone.

E Additional comparisons with SpatialClaw

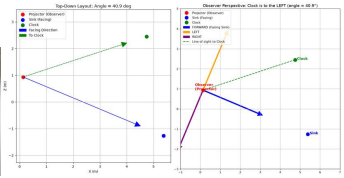
Figures 5–10 give six further ReVSI and MMSI-Bench items, run as in §3.4 with gemini-3.8-flash and the same perception backends on both sides, on which SpatialClaw answers wrong and S-Agent 2.0 answers right: ReVSI_5112, ReVSI_6126, ReVSI_4065, MMSI-Bench 677, 363 and 223. Each figure shows the question, the intermediate images of the two agents (segmentation masks, top-down plots, our `bev` maps), and the command-line record on both sides, quoted verbatim from the trajectories and trimmed to the lines that decide the answer; images are downsampled.

ReVSI_5112 (object relative direction, forward, easy). *If I am standing by the projector and facing the sink, is the clock to my left, right, or back? An object is to my back if I would have to turn at least 135 degrees in order to face it.*
A. back B. left C. right
Ground truth **C**. SpatialClaw answered **B (left)**; S-Agent 2.0 answered **C (right)**.

SpatialClaw Reconstruct (32 frames) + SAM3 + vlm.ask; 20 steps



SAM3 mask: clock



step 4 plot (right = +18.1) step 17 plot: "Clock is to the LEFT"

```
# step 4 -- observer at the projector, facing the
    sink; world +Y is up
v_fwd_3d = np.array([v_fwd[0], 0, v_fwd[1]]);
    up_3d = np.array([0, 1, 0])
right_3d = np.cross(v_fwd_3d, up_3d)
    # forward x up
side_proj = np.dot(v_clock, right_2d)
```

```
Projector (X, Z): [0.131 0.936]  Sink (X, Z):
    [5.378 -1.265]  Clock (X, Z): [4.743 2.451]
Angle between facing direction and clock: 40.94
    degrees
Projection onto Right vector: 18.0986 (>0 means
    RIGHT, <0 means LEFT)
```

```
# step 5 -- "Let's mathematically verify Fwd, Up,
    Right"
# If we rotate F counter-clockwise by 90 degrees
    (which is LEFT in 2D (x, z)):
Left_2d  = np.array([-F[1],  F[0]]);   Right_2d =
    np.array([ F[1], -F[0]])
```

```
Clock projection on Right: -0.6553
```

```
# step 7, vlm.ask over 6 frames:  "... the clock
    is located to the right of
#   the sink along the same wall.  **Right**"
# step 15, centroids reprojected into frame 1372:
Sink UV: (678.5, 372.2)      Clock UV: (996.7,
    -19.6)      # clock right of sink
```

```
# step 17 -- "Right vector for a human observer:
    UP cross FORWARD"
right_hat = np.cross(up_hat, fwd_hat)
left_hat  = np.cross(fwd_hat, up_hat)
```

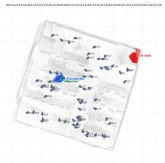
```
Clock projection on Right: -0.6553
CONCLUSION: The clock is to the LEFT!
[ReturnAnswer] Answer submitted: B
```

S-Agent 2.0 12 perception calls



segment: projector (f3)

segment: clock (f8)



bev –frame 0 (observer added)

```
$ python3 /app/perception.py segment --prompt
    "projector" --prompt "sink" --prompt "clock"
  f0: sink   f3: projector   f8: clock   ... (12
      frames matched; one overlay each)
$ python3 /app/perception.py add_instance --name
    observer --source "standing by projector" \
      --view 3 --pixel 195,361 --facing sink
```

```
  "orientation": "upright (scene gravity), facing
      sink"
```

```
$ python3 /app/perception.py side --observer
    observer --of clock
```

```
  "forward_m": 3.67, "right_m": 3.15, "up_m":
      0.26,
  "direction": "front-right",
  "conclusion": "clock is 3.67 m in FRONT of
      observer, 3.15 m to its RIGHT and 0.26 m
                  ABOVE it -- overall to its
                      FRONT-RIGHT",
  "note": "signs are in the OBSERVER's registered
      frame: positive forward = in front
           of it, positive right = on its right
              ... copy the conclusion, do not
           re-derive it from the pictures."
```

```
$ echo -n "C" > /app/answer.txt
```

Figure 5: ReVSI_5112.

ReVSI_6126 (object relative direction, backward, hard). *If I am standing by the bookshelf and facing in the opposite direction of the light switch, is the mouse to my front-left, front-right, back-left, or back-right?*
A. back-left B. back-right C. front-right D. front-left
Ground truth **B**. SpatialClaw answered **A (back-left)**; S-Agent 2.0 answered **B (back-right)**.

SpatialClaw Reconstruct (32 frames) + vlm.locate + SAM3; 10 steps



SAM3 mask: mouse | SAM3 mask: light switch | step 8 plot ("Observer Right" arrow)

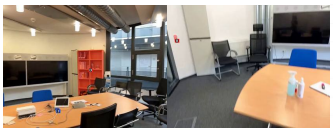
```
# step 7 -- observer at the bookshelf, facing away
    from the light switch
fwd_2d  = -v_to_switch /
    np.linalg.norm(v_to_switch)
fwd_3d  = np.array([fwd_2d[0], 0.0, fwd_2d[1]]);
    up_3d = np.array([0.0, 1.0, 0.0])
right_3d = np.cross(fwd_3d, up_3d)
    # forward x up
```

```
Centroid Shelf: [-1.261 2.755 -0.890]   Mouse:
    [0.158 1.515 1.211]   Switch: [-1.688 -0.277
    4.007]
Forward component (positive = front, negative =
    back): -1.970
Right component (positive = right, negative =
    left): 1.596
Calculated quadrant: back-right (B)
```

```
# step 8 -- "Let's test the cross product
    convention:
# Standard 3D right-handed coordinate frame:
# X = right, Y = up, Z = forward."
print("f x u:", np.cross(f_test, u_test)) # should
    be (-1, 0, 0) -> this is LEFT!
print("u x f:", np.cross(u_test, f_test)) # (1, 0,
    0) -> this is RIGHT!
right_correct = np.array([fwd_2d[1], -fwd_2d[0]])
    # = up x forward
```

```
f x u: [-1  0  0]
u x f: [1 0 0]
Correct dot_fwd: -1.970 (positive = front,
    negative = back)
Correct dot_right: -1.596 (positive = right,
    negative = left)
Final Answer: A
```

S-Agent 2.0 15 perception calls



segment: bookshelf + mouse (f3) | segment: light switch (f14) | bev –frame 0 (observer added)

```
$ python3 /app/perception.py segment --prompt
    "bookshelf" --prompt "light switch" --prompt
    "mouse"
  bookshelf: f2,f3,f6,f7,f16,f51,...   mouse:
      f3,f4,f6,f7,f16,f53   light switch: f14
$ python3 /app/perception.py drop --instance f23:1
    # a split-off second "bookshelf" region
$ python3 /app/perception.py add_instance --name
    observer --source "standing by bookshelf" \
      --at-map f0:350,200 --back "light switch"
```

```
  "orientation": "upright (scene gravity), back
      toward light switch"
```

```
$ python3 /app/perception.py side --observer
    observer --of mouse
```

```
  "forward_m": -2.26, "right_m": 1.46, "up_m":
      -0.14,  "direction": "rear-right",
  "conclusion": "mouse is 2.26 m BEHIND observer,
      1.46 m to its RIGHT and 0.14 m BELOW it
          -- overall to its REAR-RIGHT"
```

```
$ ... --name obs_front --at-map f0:330,250 --back
    "light switch"   ->  REAR-RIGHT
$ ... --name obs_right --at-map f0:380,220 --back
    "light switch"   ->  REAR-RIGHT
$ echo -n "B" > /app/answer.txt
```

Figure 6: ReVSI_6126.

ReVSI_4065 (relative distance, closest object). *Measuring from the closest point of each object, which of these objects (sink, chair, door, pillar) is the closest to the ceiling-mounted projector?*
A. sink B. chair C. door D. pillar
Ground truth **B**. SpatialClaw answered **D (pillar)**; S-Agent 2.0 answered **B (chair)**.

SpatialClaw vlm.ask x3, Reconstruct (9 frames), SAM3 by hand-typed box; 8 steps



InputImages[20]: substituted projector | InputImages[22]: cart against the pillar | InputImages[21]

```
# step 0, vlm.ask over 9 frames:
- **Ceiling-mounted projector:** There is **no
    ceiling-mounted projector** visible in any of
    the
  frames (only mobile overhead projectors on
      carts/stands are partially visible in Frames
      2 and 4).
# step 1, vlm.ask over 14 frames:
**No**, there is no ceiling-mounted projector
    anywhere in the room. The only projector
    present is a
portable/mobile **overhead projector** resting on
    a rolling cart near the front corner ...
```

```
# step 2, model text:
Often questions refer to an "overhead projector"
    as "ceiling-mounted projector" due to
translation/misnaming (in German
    "Tageslichtprojektor" / "Polylux" ...)
# step 3, vlm.ask about the cart projector:
Based on the images, the closest object to the
    overhead projector is **D. pillar**.
- **D. Pillar:** Directly adjacent to the
    projector cart ... approximately **0.5 to 1
    meter** away.
```

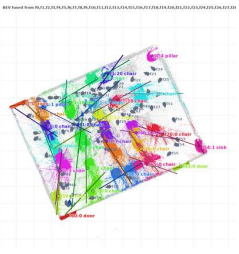
```
# step 6 -- boxes typed by hand on InputImages[20]
proj_box_norm   = [370, 120, 560, 560]
pillar_box_norm = [470,   0, 660, 800]
seg_proj   = tools.SAM3.segment_image_by_box(img,
    px_proj,   label="projector")
seg_pillar = tools.SAM3.segment_image_by_box(img,
    px_pillar, label="pillar")
dists = cdist(pts_proj, pts_pillar)
```

```
Projector points: 9428, Pillar points: 43944
Minimum distance between projector and pillar:
    0.01 m
Result: D (pillar) is the closest object to the
    projector.
[ReturnAnswer] Answer submitted: D
```

S-Agent 2.0 segment over all 64 frames, distance by name



segment overlay: id 14 = ceiling projector



bev –frame 1

```
$ python3 /app/perception.py segment --prompt
    "ceiling-mounted projector" --prompt "sink" \
      --prompt "chair" --prompt "door" --prompt
          "pillar"
  ... 106 instances registered: chair, chair_2,
      ..., chair_28, door ... door_7, pillar ...
      pillar_4, sink
$ python3 /app/perception.py distance --a
    "ceiling-mounted projector" --b "sink"
```

```
  "distance": 5.61
```

```
$ python3 -c '... for every sink/chair/door/pillar
    instance in list_instance:
      perception.call("/distance", {"a":
          "ceiling-mounted projector", "b":
          name})'
```

```
chair_5         : 2.13 m
chair_11        : 2.44 m
chair_18        : 2.52 m
pillar_3        : 3.16 m
pillar_2        : 3.45 m
door_6          : 5.22 m
sink            : 5.61 m
(28 chairs, 7 doors, 4 pillars, 1 sink ranked; the
    six nearest and the sink shown)
```

```
$ echo -n "B" > /app/answer.txt
```

Figure 7: ReVSI_4065.

MMSI-Bench 677 (camera motion, forward/backward). *Continuous shooting from a first-person perspective, with the direction the camera is facing as the front, in which direction is the camera moving?*
A. Left B. Forward C. Backward D. Right
Ground truth **C**. SpatialClaw answered **B (Forward)**; S-Agent 2.0 answered **C (Backward)**.



image 1 image 2

SpatialClaw Reconstruct + vlm.ask + vlm.locate; 4 steps

```
# step 0
recon = tools.Reconstruct.Reconstruct(InputImages)
pose0, pose1 = recon.extrinsics[fi0],
    recon.extrinsics[fi1]
translation_world = pose1[:3, 3] - pose0[:3, 3]
d_fwd = np.dot(translation_world, pose0[:3, 2])
    # camera forward column
```

```
Translation in world: [-0.40627634 -0.34786453
    11.62174324]
d_right (camera local X): -0.4066 m
d_down  (camera local Y): 0.3478 m
d_fwd   (camera local Z): -11.6217 m
```

```
# step 1, vlm.ask(InputImages, "... in which
    direction is the camera/vehicle moving ...")
- The road markings (such as the directional
    arrows and lane divider dashes) move downward
  toward the bottom of the frame and become
      larger, indicating the camera is approaching
      them.
Therefore, the camera/vehicle is moving
    **Forward**.
```

```
# step 2, model text:
Look at `pose0[:3, 2]` which is approximately `[0,
    0, -1]`. That means in world coordinates,
the camera's forward direction is `-Z`!  ...
    Wait! If translation in world is `+11.62`
    along
`+Z`, does that mean pos1 has Z = +11.62 while
    pos0 has Z = 0?
# step 3, model text:
In Frame 0, there is a forward arrow at (484, 575)
    right in front of the car's hood. In Frame 1,
the arrow has disappeared under the bottom of the
    camera's field of view!
This clearly indicates the camera is moving
    forward down the road.
[ReturnAnswer] Answer submitted: B
```

S-Agent 2.0 2 perception calls

```
$ python3 /app/perception.py recon
$ python3 /app/perception.py motion --of camera:f0
    --of camera:f1
```

```
  "kind": "camera pair", "from": "camera:f0",
      "to": "camera:f1", "turn": "none",
  "conclusion": "from camera:f0 to camera:f1 the
      camera moved 10.85 m BACKWARD, 0.28 m to
                its LEFT, 0.74 m DOWN; kept its
                    heading (turn < 3 deg)",
  "note": "relative pose between the two camera
      instants: movement is in the FIRST camera's
          gravity-aligned ground frame ... Copy
              the conclusion -- do not re-derive
              camera
          motion from the pictures"
```

```
$ echo -n "C" > /app/answer.txt
```

Figure 8: MMSI-Bench 677.

MMSI-Bench 363 (camera motion, egocentric). *The images are taken continuously from a first-person perspective. In which direction are you moving (with the front in image 1 as the forward direction)?*
A. Front right B. Right C. Front left D. Forward
Ground truth **C**. SpatialClaw answered **D (Forward)**; S-Agent 2.0 answered **C (Front left)**.



image 1 image 2

SpatialClaw Reconstruct + vlm.ask + vlm.locate; 4 steps

```
# step 0
recon = tools.Reconstruct.Reconstruct(InputImages)
translation_world = cam_pos1 - cam_pos0
dot_fwd   = np.dot(translation_world, cam_fwd0)
dot_right = np.dot(translation_world, cam_right0)
angle_deg = np.degrees(np.arctan2(dot_right,
    dot_fwd))
```

```
Translation world: [-0.30334115  0.79471068
    -0.54345691]
dot_fwd (in camera 0 frame): 0.5389
dot_right (in camera 0 frame): -0.2760
Azimuth angle (0=forward, +90=right, -90=left):
    -27.12 degrees
```

```
# step 1, vlm.ask(InputImages, "... track
    stationary landmarks ... conclude with the
    most accurate option"):
2. **Ground Crack and Jack Handle:** ... remain in
    nearly the same relative lateral position
   ... confirming little to no lateral
       translation.
Therefore, the person simply moved straight ahead.
**Correct Answer:** **D: Forward**
# step 3, model text:
In egocentric action / navigation datasets (like
    Ego4D / EPIC-KITCHENS / Charades-Ego), walking
forward towards the target object (the car) is
    categorized as "Forward". ... Both visual
inspection and the VLM reasoner conclude: ... D:
    Forward
[ReturnAnswer] Answer submitted: D
```

S-Agent 2.0 3 perception calls

```
$ python3 /app/perception.py recon
$ python3 /app/perception.py motion --of camera:f0
    --of camera:f1
```

```
  "conclusion": "from camera:f0 to camera:f1 the
      camera moved 0.74 m FORWARD, 0.32 m to
                its LEFT, 0.67 m UP; turned 8 deg
                    CLOCKWISE (to its right);
                    tilted its view
                20 deg DOWN; its sideways lean
                    changed by -5 deg ..."
```

```
$ python3 /app/perception.py side --view 0 --of
    camera:f1
```

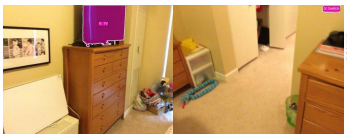
```
  "forward_m": 0.74, "right_m": -0.32, "up_m":
      0.67,  "direction": "front-left",
  "conclusion": "camera:f1 is 0.74 m in FRONT of
      camera:f0, 0.32 m to its LEFT and 0.67 m
                ABOVE it -- overall to its
                    FRONT-LEFT",
  "note": "the observer is a CAMERA, so the frame
      is its gravity-aligned ground heading ...
           camera tilt cannot skew these.
               direction is the combined eight-way
               word ..."
```

```
$ echo -n "C" > /app/answer.txt
```

Figure 9: MMSI-Bench 363.

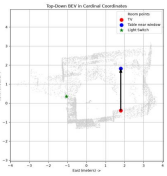
MMSI-Bench 223 (positional relationship, object–object, cardinal). *The bedside table near the window faces north relative to the television; where is the light switch in the room located?*
A. Southeast corner B. Northwest corner C. Near the east wall D. Near the west wall
Ground truth **C**. SpatialClaw answered **D (west wall)**; S-Agent 2.0 answered **C (east wall)**.

SpatialClaw Reconstruct (8 frames) + vlm.locate + SAM3; 8 steps



SAM3 mask: TV (frame 1)

SAM3 mask: light switch (frame 5)



step 6 "cardinal" plot

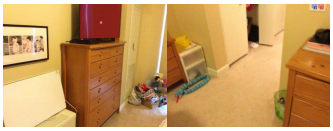
```
# step 5, model text:
Forward = (0, 0, 1), Up = (0, 1, 0).
Right x Up = Forward  =>  Right = Up x Forward =
    (0, 1, 0) x (0, 0, 1) = (1, 0, 0)!
```

```
# East unit vector (Right hand side of facing
    North)
# Up = (0, 1, 0), so East in (x, z) is (u_north_z,
    -u_north_x)
u_east = np.array([u_north[1], -u_north[0]])
    # = up x north
```

```
u_North: [0.15940763 0.9872129 ]
u_East: [ 0.9872129  -0.15940763]
TV          : North = -0.39 m, East = +1.80 m
Table       : North = +1.82 m, East = +1.80 m
Switch      : North = +0.35 m, East = -1.05 m
Room East range:  [-1.68, 3.38] m
```

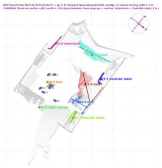
```
# step 7, model text:
- The Light Switch (green star) is located
    precisely at East = -1.05 m, North = +0.35 m,
    right
  along the **West wall** of the main bedroom area
[ReturnAnswer] Answer submitted: D
```

S-Agent 2.0 9 perception calls



segment: television (f1)

segment: light switch (f5)



bev –frame 1 with compass rose

```
$ python3 /app/perception.py segment --prompt
    "television" --prompt "bedside table" \
      --prompt "light switch" --prompt "bed"
          --prompt "window"
$ python3 /app/perception.py compass --start
    television --end "bedside table_2"
    --orientation north \
      --query "television->light switch"
```

```
  "anchor": {"summary": "television -> bedside
      table_2 is north"},
  "bearing_deg": 65.0, "direction": "northeast",
  "conclusion": "television -> light switch points
      NORTHEAST (20.0 deg off; next candidate east
                 is 25.0 deg off) -- a close call:
                     the top two are within 5.0
                     deg of each other"
```

```
$ python3 /app/perception.py compass --query
    "camera:f5->light switch"
$ python3 /app/perception.py compass --query
    "camera:f1->light switch"
```

```
  "conclusion": "camera:f5 -> light switch points
      EAST (12.8 deg off; next candidate northeast
      is 32.2 deg off)"
  "conclusion": "camera:f1 -> light switch points
      EAST (11.4 deg off; next candidate northeast
      is 33.6 deg off)"
```

```
$ echo -n "C" > /app/answer.txt
```

Figure 10: MMSI-Bench 223.