CAPAGENT: KEEPING SPATIAL AGENTS RELIABLE WHEN THEIR PERCEPTION MODELS CHANGE

Anonymous authors
Paper under double-blind review

ABSTRACT

Spatial embodied agents rely on external perception models for depth, pose, and grounding to make task decisions, and these models are often replaced after the agent is trained. Similar readings from different models can differ in how far they can be trusted, so the right action can change after a replacement. This paper asks how one trained agent can make correct decisions with unseen perception models under fixed task rules and interfaces, without updating its weights. The agent judges reliability from quality cues, such as how widely depth values spread around a doorway, but a cue rises both at real scene edges and when the model is noisier. We show that an agent trained on one output at a time, from many scenes but few models, learns mostly the scene-driven meaning of these cues and underreacts when a new model raises them. We first build CAPSWAP-BENCH, a benchmark that replays actual model replacements on matched scenes, with switch pairs where the right action changes and retain pairs where it should not. We then present CAPAGENT, which trains on outputs of two perception models for the same scene and query, so that their difference isolates the model change. The core idea is to learn how its decision margins should move within such pairs, crossing a task threshold only when the right action changes; at deployment, CAPAGENT still reads one output at a time. When three tool models are swapped for unseen ones, CAPAGENT-8B reaches an average score of 70.6 on five standard spatial benchmarks, retains 96% of its original-tool score, and matches the 27B SpatialCLI agent. On CAPSWAP-BENCH, it acts correctly on 84.7% of switch pairs and 92.3% of retain pairs. Anonymous code repository: `https://anonymous.4open.science/r/CAPAgent-8E4C/`.

1 INTRODUCTION

Spatial embodied agents call external perception models as tools (for depth estimation, camera pose, and visual grounding) and use their outputs to make high-level task decisions, such as whether to proceed, stop, observe again, or reuse a cached view (Surís et al., 2023; Chen et al., 2026; Zhou et al., 2026). In practice, engineers often replace these perception tools after the agent is trained, to meet memory, latency, or accuracy requirements (Wu et al., 2025; Hindsbo et al., 2026; Yang et al., 2024). A new tool keeps the output format but may differ in reliability. This creates a key requirement: the agent's task reasoning should not depend on the specific noise and reliability characteristics of the tools it was trained with. Given fixed task rules and unchanged tool interfaces, can one trained agent make correct decisions with unseen perception tools, without updating its policy?

The core challenge is that the same reading can be more or less trustworthy depending on the tool that produced it. A depth tool reports a doorway clearance of 1.19 m, and the robot needs 1.15 m. If the plausible range of the true clearance is narrow, it lies above 1.15 m and the robot can go through; if it is wide, it crosses 1.15 m and the robot should measure again (Figure 1). The scene and requirement are the same; only the tool differs. Conversely, two precise tools that read 1.19 m and 1.22 m both support going through. The agent never sees this range; it sees only the reading and a few quality cues, such as the spread of depth values around the doorway and the fraction of valid pixels. After a swap, it must use these cues to change decisions that lose support and keep the rest.

Standard tool-use training, by supervised fine-tuning or outcome-based reinforcement learning, shows the agent one output at a time from its training tools (Qin et al., 2024; Chen et al., 2026;

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.

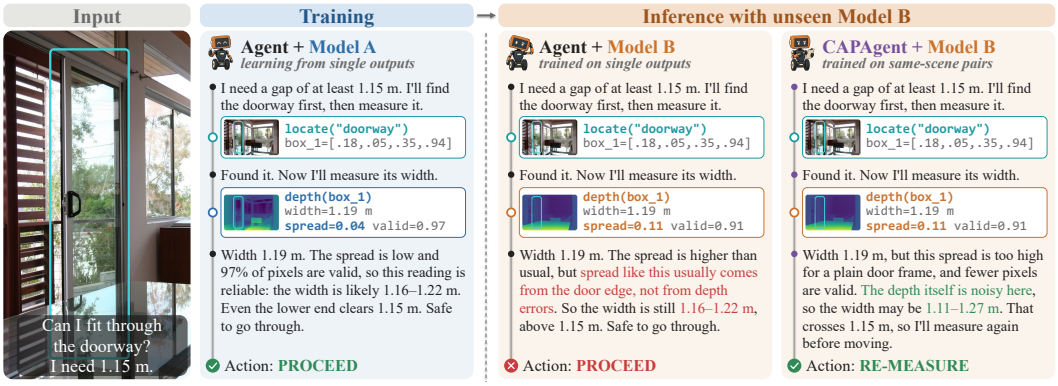


Figure 1: **After a model swap, the same width reading can call for a different action.** Unseen model B reports the same width as model A with a wider spread. The single-output agent treats the spread as a door edge and proceeds; CAPAGENT treats it as depth noise, finds that the range crosses 1.15 m, and measures again.

Zhou et al., 2026; Shao et al., 2024). This does not teach the agent what a cue means after a swap, because a cue can rise for two reasons. Depth values spread more around a door frame with a real edge, and also when the depth model is noisier. A real edge raises spread with little error in the reading; model noise raises both. On our benchmark, the calibrated error radius rises about three times as steeply with spread when the model changes on a fixed scene as when the scene changes under a fixed model (Section 5.5). Training data covers many scenes but few models, so most cue variation comes from scenes. The agent learns the weak, scene-driven meaning of the cue and underreacts when a new model raises it. It can do well with its training tools yet fail after a swap.

Addressing this problem requires two things: a controlled test of model replacement, and a training signal that separates model changes from scene changes. A controlled test must change only the perception model, keeping the scene, query, task rule, and tool interface fixed, and it must know the right action for each model's output. Existing tool-use benchmarks test tool calling, also with noisy tool documentation (Patil et al., 2025; Lu et al., 2025; Ye et al., 2024) and keep the model behind each tool fixed. We therefore build CAPSWAP-BENCH, which runs real depth, pose, and grounding models on the same scenes and queries, calibrates each model's error radius on separate scenes with ground truth, and turns every output into a reference action (Figure 2). Two outputs for one scene form a *switch* pair if the right action changes and a *retain* pair if it stays the same.

The same principle tells us how to train: compare outputs in which only the model changes. We run two perception models on the same image and query and show the agent both outputs, one at a time. The scene is shared, so the difference between the outputs isolates what the swap did to the evidence. An action label cannot say how far a decision should move between the two outputs, so CAPAGENT predicts, along with its response, *decision margins*: signed distances from the plausible range of a measurement to each task threshold, from which a fixed rule reads the action. On same-scene pairs, training fits how the margins change, pushes them across a threshold when the action should switch, and keeps the margins of capabilities whose tool did not change. At deployment the weights stay fixed, and the agent reads one anonymous output at a time. Our contributions are:

- A linear model (Proposition 1) and agent probes that show why single-output training transfers poorly: cues mix scene and model variation, and the agent learns the scene-driven meaning.
- CAPSWAP-BENCH, which tests real replacements among 15 depth, pose, and grounding models on matched scenes, with switch and retain pairs and ground-truth outcomes.
- CAPAGENT, which learns decision-margin changes from same-scene pairs and reads one output at a time. With unseen tools, CAPAGENT-8B keeps 96% of its standard-benchmark score, level with SpatialCLI-27B, and acts correctly on 84.7% of CAPSWAP-BENCH switch pairs.

2 RELATED WORK

Spatial agents and model updates. Language models learn to call tools from annotated or generated traces (Yao et al., 2023; Schick et al., 2023; Qin et al., 2024; Patil et al., 2024), and visual

programs and spatial agents bring depth, pose, and grounding models into this loop (Gupta & Kembhavi, 2023; Surís et al., 2023; Chen et al., 2026; Zhou et al., 2026). These systems are trained and evaluated with a fixed set of perception models. RoTBench perturbs tool and parameter names in the tool documentation (Ye et al., 2024); we keep the interface and replace the model behind it. Backward-compatible and positive-congruent training keep downstream behavior stable after a model update (Shen et al., 2020; Yan et al., 2021; Bansal et al., 2019; Srivastava et al., 2020), whereas our agent must sometimes change its decision; test-time adaptation updates a model on test data (Wang et al., 2021), whereas ours stays frozen.

Judging reliability. Selective prediction and conformal methods calibrate the deployed model (Geifman & El-Yaniv, 2017; Angelopoulos & Bates, 2023; Ren et al., 2023); introspective perception and depth uncertainty learn a known model's errors (Daftry et al., 2016; Poggi et al., 2020); accuracy prediction under shift uses unlabeled target data (Garg et al., 2022). We judge an unseen model from one output.

Invariance and learning from pairs. Domain generalization methods seek a predictor that is optimal in every training environment (Arjovsky et al., 2019), similar risks across environments (Krueger et al., 2021), or strong worst-group performance (Sagawa et al., 2020); domain-adversarial training removes domain information from the representation (Ganin et al., 2016). With perception models as environments, these methods constrain the representation or the risks across models, but none supervises how a decision should change between two outputs of one scene. Comparing two measurements of one unit cancels a shared effect, as in fixed-effects models (Wooldridge, 2010); counterfactual pairs, contrast sets, and relational distillation also learn from differences (Kaushik et al., 2020; Gardner et al., 2020; Park et al., 2019). Gradient aligned regression (GAR) fits prediction differences to label differences over all pairs of a batch (Zhu et al., 2025). We pair only outputs that differ in the provider, and tie the change to a decision rule.

3 Problem Formulation

Tools, packets, and decisions. We study three decisions that depend on a perception model: *passage* (is this doorway wide enough?), *cache reuse* (is a cached view still valid?), and *object selection* (which object matches the query?). For scene x and task τ, the model behind the tool, which we call the *provider* P, returns a *packet* $e = P(x, \tau)$: the measurement plus a few quality cues around the query region. For depth these are the spread of depth values, the fraction of valid pixels, and the edge energy of the image crop, which is the same for every provider. The agent chooses an action $a = \pi(e, \tau)$. A *swap* changes P and keeps the scene, task, packet format, and agent weights.

State, margin, and action. A *margin* is the signed distance from an end of the plausible range of a measurement to a decision threshold, divided by a positive task scale. The *decision state* ϕ is the vector of a task's margins. A fixed rule reads the discrete *action* from the signs of the state, $a^\star = q(\phi)$. For a depth estimate p with error radius r, required clearance θ, and scale $s_\tau > 0$,

$$\phi_{\text{depth}} = \left[\frac{p - r - \theta}{s_\tau}, \quad \frac{p + r - \theta}{s_\tau} \right]. \tag{1}$$

Two positive margins mean the whole range clears the requirement (`PROCEED`), two negative margins mean it does not (`STOP`), and mixed signs mean the range crosses it (`RE-MEASURE`). With $p = 1.19\,\text{m}$ and $\theta = s_\tau = 1.15\,\text{m}$, a radius of $0.03\,\text{m}$ gives margins $(+0.009, +0.061)$ and `PROCEED`; $0.08\,\text{m}$ gives $(-0.035, +0.104)$ and `RE-MEASURE`. Pose and grounding follow the same pattern (Appendix E.2).

Reference actions and deployment setting. Each provider is calibrated on separate scenes with ground truth: a residual regressor reads its full output and image features, and a split-conformal step sets r at a nominal 90% level (Angelopoulos & Bates, 2023); held-out coverage ranges from 88.9% to 91.1% across providers. The rule then gives a reference action for every output. The reference sees more than the agent: its action is the one the calibrated evidence supports, and we use it as the training target and for scoring. At deployment the agent reads one packet at a time, without the provider's name, with fixed weights, and with no labels for a new provider.

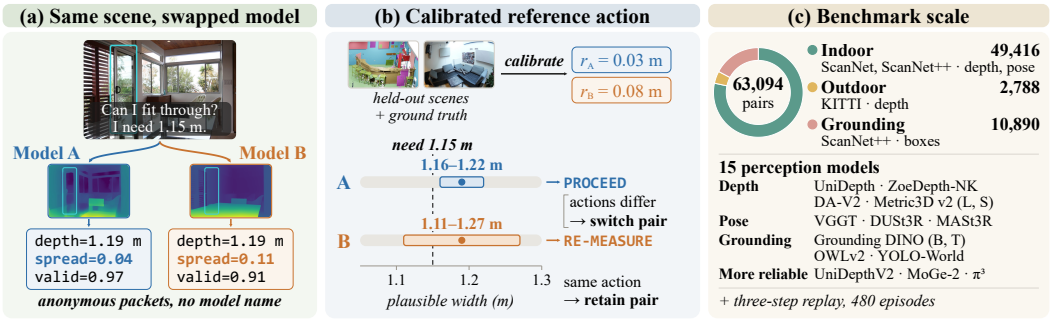


Figure 2: **Only the provider changes within a pair.** (a) Two depth models process the same scene and query; the agent reads each anonymous packet alone. (b) Each model's radius r is calibrated on held-out scenes; $1.19 \pm r$ m against 1.15 m gives the reference action, and the pair is a switch pair when the two actions differ. (c) Tracks and the 15 providers.

Benchmark. CAPSWAP-BENCH builds comparisons in which only the provider changes (Figure 2). The indoor pool combines 1,513 ScanNet scans and 906 ScanNet++ scenes (Dai et al., 2017; Yeshwanth et al., 2023), split into 100 capture units for calibration, 200 for training, 60 for validation, and 2,059 for evaluation, with rescans of one space kept in one split. Fifteen providers cover depth, pose, and grounding, including three more reliable than any training provider (Appendix D). The *controlled core* pairs the reference depth and pose providers with six less reliable replacements on 510 evaluation units: 3,060 pairs, 816 of which change the reference action. Each record holds one depth and one pose query of a unit; a pair swaps one provider. The full indoor set (49,416 pairs), outdoor, grounding, and replay tracks test breadth.

Metrics. Two metrics are common: **Acc**, agreement with the reference action over all decisions, and **Unsup.**, the rate of unsupported commitments (`PROCEED`, `USE-CACHE`, or `ACT` when the reference does not support committing). Two are specific to swaps, and each catches one way to fail. **Switch** is accuracy on pairs whose reference action differs; an agent that ignores the swap is right on only one output of such a pair. **Retain** is accuracy on pairs whose reference action is the same; an agent that reacts to every change in the cues fails it. Both average the two outputs of a pair. *Unseen* restricts Switch to pairs whose replacement was absent from training.

4 METHOD

4.1 PREDICTING THE DECISION STATE FROM ONE OUTPUT

An action label says on which side of a threshold the evidence lies, not how far, and this distance is what a swap moves. We therefore train the agent to predict margins. The response starts with one anchor token per capability. A shared two-layer network g_ψ reads the anchor's hidden state $\mathbf{h}_c$, and a capability-specific projection W_c gives the center and half-width of each plausible range, with a softplus keeping the half-width positive; $\hat{\phi}_c$ is the center minus and plus the half-width. A masked Smooth-L_1 loss fits the reference margins, $L_{\text{state}} = \sum_j m_j \ell(\hat{\phi}_j, \phi_j) / \sum_j m_j$, where m_j marks the coordinates that exist for the task; q then reads the action without generating a response.

4.2 WHY SAME-SCENE PAIRS HELP

Take one depth cue z, such as spread, and write $z = s + b$: s is the part set by the scene and b the part set by the provider. The error radius follows $r = \alpha s + \beta b + \epsilon$, so α and β say how strongly each part goes with error; a real edge raises spread with little extra error, which we model as $\alpha < \beta$. The image cue u (edge energy) is the same for every provider and explains part of the scene term, $s = tu + v$. Consider a predictor $\hat{r} = kz + hu + c_P$ with a per-provider offset c_P.

Proposition 1 (Pairs set the response to provider-driven cue changes)**.** *Let σ_v^2 be the scene variance that u does not explain and σ_w^2 the variance of b within a provider. Under the independence assumptions of Appendix B, minimizing the squared error on both outputs of a same-scene pair plus $\lambda \geq 0$*

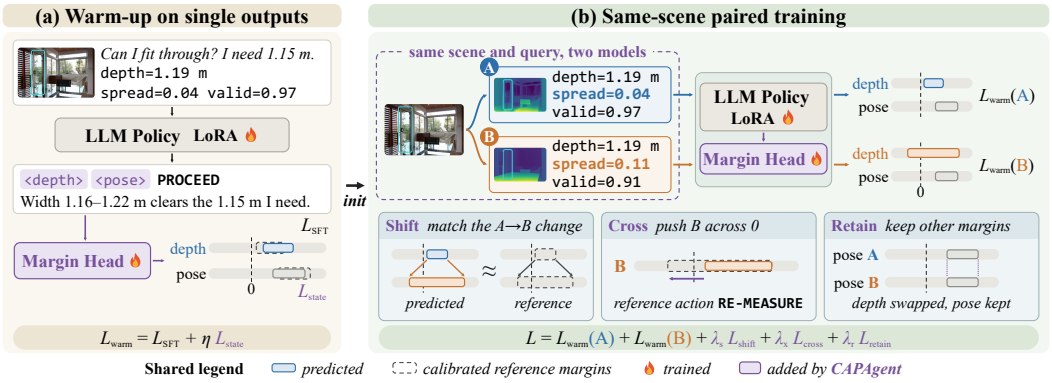


Figure 3: **Two-stage training of CAPAGENT.** (a) Warm-up on one output: L_{SFT} fits the response, and the margin head fits the reference margins from the anchor states (L_{state}). (b) The two outputs of one scene pass separately through shared weights; shift, cross, and retain act on their margins. Purple: what CAPAGENT adds; only LoRA and the margin head are trained.

times the squared error on their difference gives $h = (\alpha - k)t$ and

$$k(\lambda) = (1 - w_\lambda)\,\alpha + w_\lambda\,\beta, \qquad w_\lambda = \frac{(1+\lambda)\sigma_w^2}{\sigma_v^2 + (1+\lambda)\sigma_w^2}. \tag{2}$$

The learned response to a cue change within a pair is $k(\lambda)$, and the response across scenes is $k(\lambda) + (\alpha - k(\lambda))\rho$ with $\rho = t^2\mathrm{Var}(u)/\mathrm{Var}(z)$. Pairing independent scenes gives $k(0)$ for every λ.

Single-output training ($\lambda = 0$) weights the two meanings by σ_v^2, the scene variation left after the image cue, and σ_w^2; when the first is larger, $k(0)$ lies near α and the agent underreacts to a swap. Three predictions follow. Same-scene pairs move k toward β. Pairs from different scenes leave k at $k(0)$; a squared pairwise loss over all outputs of a batch equals the variance of the pointwise errors (Zhu et al., 2025, Thm. 1) and stays near $k(0)$ when the batch spans many scenes. The image cue separates the two responses: without it ($t = 0$) they are equal, and the optimal image weight $h = (\alpha - k)t$ grows in magnitude with k, so a stronger response to model-driven changes calls for more weight on the image cue. The proposition gives every provider the same α and β; this shared link between cue and error is what carries to a new provider, and Section 5.6 tests one that breaks it.

4.3 LEARNING DECISION CHANGES FROM PAIRS

A pair holds outputs $e_A = P_A(x, \tau)$ and $e_B = P_B(x, \tau)$ of two providers on one scene and task, read in separate forward passes with shared weights. Let $d = \phi_B - \phi_A$ be the reference change in the state and $\hat{d} = \hat{\phi}_B - \hat{\phi}_A$ the predicted change. The shift loss matches the two on the set J of margins that belong to the swapped capability:

$$L_{\mathrm{shift}} = \frac{1}{|J|} \sum_{j \in J} \ell(\hat{d}_j, d_j). \tag{3}$$

The scene is shared, so d holds only what the provider changed. We fit the signed change in task units, because its size decides whether it reaches a threshold. Two terms connect the change to the rule. A change of about the right size can stop short of a threshold, so a *cross* term asks the new margins to land on the correct side when the action should switch. Capabilities share a policy and head, and the cache check takes the scene range from the depth packet, so a depth swap can also move the pose prediction; a *retain* term, distinct from the retain pairs of CAPSWAP-BENCH, keeps it in place. Let S hold the capabilities whose reference action changes, U the coordinates whose reference margins stay the same (the query whose provider did not change), and $\tilde{q}_c$ the soft version of q with each sign test replaced by a sigmoid. Then

$$L_{\mathrm{cross}} = \sum_{c \in S} \mathrm{CE}\big(\tilde{q}_c(\hat{\phi}_{B,c}), a^\star_{B,c}\big), \qquad L_{\mathrm{retain}} = \sum_{j \in U} |\hat{d}_j|, \tag{4}$$

and $L_{\mathrm{pair}} = \lambda_s L_{\mathrm{shift}} + \lambda_x L_{\mathrm{cross}} + \lambda_r L_{\mathrm{retain}}$.

From margins to actions. We supervise margins but report actions. The error in a new margin equals the error in the starting margin plus the error in the predicted change, $\hat{\phi}_B - \phi_B = (\hat{\phi}_A - \phi_A) + (\hat{d} - d)$, so a margin has the right sign whenever

$$|\hat{\phi}_{A,j} - \phi_{A,j}| + |\hat{d}_j - d_j| < |\phi_{B,j}|, \tag{5}$$

and the action is right when this holds for every margin the rule reads. Near a threshold $|\phi_{B,j}|$ is small and the change must be accurate; far from it a rough change is enough (Table 4b).

Training and deployment. Warm-up (Figure 3a) minimizes $L_{\text{warm}} = L_{\text{SFT}} + \eta L_{\text{state}}$ on single outputs; the pair stage (Figure 3b) minimizes $L_{\text{warm}}(A) + L_{\text{warm}}(B) + L_{\text{pair}}(A, B)$ on same-scene pairs. We train LoRA adapters (Hu et al., 2022) and the margin head. At deployment the agent reads one packet and returns $q(\hat{\phi})$. In a loop, it states a check with each tool call, which fixes the measurement, threshold, and scale; from the returned values, the margins decide whether to use the reading, query again, or answer without it (Algorithm 1).

5 Experiments

5.1 Setup

Benchmarks and swap. We follow SpatialCLI's evaluation on SpatialCLI-Bench, MindCube, MMSI-Bench (camera motion and camera position), DA-2K, and BOPASK (trajectory and object rearrangement), and report its macro average (Zhou et al., 2026; Wang et al., 2026a; Yang et al., 2026; 2024; Bhat et al., 2026). The swapped setting replaces Depth Anything 3 with Metric3D v2 Small, VGGT with MASt3R for camera motion, and Grounding DINO with YOLO-World-L (Lin et al., 2026; Hu et al., 2024; Wang et al., 2025a; Leroy et al., 2024; Liu et al., 2024; Cheng et al., 2024), keeping each tool's format; none is in our training pool.

Training. Unless noted, the 8B agents we train are LoRA adapters on Qwen3-VL-8B-Instruct (Bai et al., 2025), the base model of SpatialCLI-8B (Zhou et al., 2026). The controlled indoor policy trains on 200 capture units with UniDepth and VGGT as references and ZoeDepth-NK and DUSt3R as treatments, the replacements seen in training; the agents of Table 1 add 2,000 check-format examples and SpatialCLI's released tool-use trajectories; outdoor and grounding use separate policies. Validation uses training providers only; 95% bootstrap intervals resample physical spaces on CapSwap-Bench and questions on the standard benchmarks (Appendix C).

Baselines. *Single-output*: trajectory SFT (Qin et al., 2024), SFT + GRPO (Shao et al., 2024), and the single-output margin baseline (CAPAgent without pair losses). *Domain generalization*: DANN, which removes provider information from the anchor state (Ganin et al., 2016), and V-REx, which equalizes risks across providers (Krueger et al., 2021). *Paired*: paired-input SFT, which reads both packets; DPO on the right and wrong action of a switch pair, with our margin head (Rafailov et al., 2023); and GAR with its published objective on our margins (Zhu et al., 2025). A fixed-radius rule uses one radius per task (Appendix F).

5.2 Main results on standard spatial benchmarks

A swap hurts most where one measurement decides the answer. SpatialCLI-8B loses 6.7 points on average (Table 1): 12.0 on SpatialCLI-Bench and 12.9 on camera position (Table 14), where one depth or pose reading often settles the question, and under one point on BOPASK. Tool-use SFT-8B, trained on the check format, still loses 5.9, so the format explains little of the drop.

CAPAgent keeps its score after the swap. CAPAgent-8B keeps 96% of its score (73.4 to 70.6), 0.2 points above SpatialCLI-27B and the best swapped-tool average in Table 1, and is the best 8B agent on six of the seven columns. Its lead is largest where the swap hurts most: 86.2 on SpatialCLI-Bench and 48.4 on camera position, against 79.3 and 40.9 for SpatialCLI-8B. Single-output-8B has the same base model, data, cues, checks, and margin head, so the 2.7-point gap [1.8, 3.6] measures what the pair losses add; the gap to GAR-8B is 1.8 [1.0, 2.6] (paired bootstrap over questions). Camera motion is the exception (37.8 against 38.1 for GAR-8B): for small camera

Table 1: **Standard spatial benchmarks after a tool swap** (%, SpatialCLI's loop; Depth Anything 3, VGGT, and Grounding DINO replaced by unseen models). Average: macro average with swapped and original tools (Table 14). Frontier original-tool scores are SpatialCLI's reported runs, without our quality fields. Bold / underline: best / second among 8B agents.

	SpatialCLI		MMSI		BOPASK			Average		
Agent	Bench	MindCube	Motion-Cam	Pos-Cam-Cam	DA-2K	Traj.	ObjRrr	Swapped	Original	Change
Frontier models, prompted with SpatialCLI tools										
GPT-5.6 Sol	65.2	69.8	52.4	55.9	80.6	52.6	49.1	64.1	68.1	−4.0
Qwen3.5-397B-A17B	70.1	64.0	43.9	45.6	85.3	57.4	44.2	63.0	68.2	−5.2
Larger open agent										
SpatialCLI-27B	81.9	81.0	49.6	43.8	85.9	57.8	54.9	70.4	75.9	−5.5
8B agents										
SpatialCLI-8B	79.3	79.0	35.1	40.9	85.2	54.8	47.6	66.5	73.2	−6.7
Tool-use SFT-8B	80.6	79.4	35.1	41.8	85.9	54.8	47.7	67.1	73.0	−5.9
Single-output-8B	81.8	79.9	36.5	43.0	86.6	54.9	47.8	67.9	72.9	−5.0
GAR-8B	83.1	80.6	**38.1**	44.6	87.5	55.0	47.9	68.8	73.1	−4.3
CAPAGENT-8B (ours)	**86.2**	**82.3**	37.8	**48.4**	**89.4**	**55.2**	**48.6**	**70.6**	**73.4**	**−2.8**

Table 2: **Controlled model swaps on the CAPSWAP-BENCH core** (3,060 pairs, 816 switch). Bold / underline: best / second.

Method	Trained on	Acc↑	Unsup.↓	Switch↑	Unseen↑	Retain↑
Trajectory SFT	actions	78.1	5.6	54.1	41.3	86.8
SFT + GRPO	actions, reward	79.3	5.1	57.6	44.0	87.2
Single-output margin	margins	83.5	4.1	68.4	52.8	89.0
DANN	margins, provider-adversarial	83.7	4.0	70.3	55.2	88.6
V-REx	margins, providers as environments	84.2	3.9	71.2	56.0	88.9
Paired-input SFT	both actions	80.4	5.0	63.9	48.6	86.4
DPO + margin head	action preferences, margins	83.4	4.9	74.6	58.4	86.6
GAR	margin differences, batch pairs	85.5	3.7	73.9	59.1	89.7
CAPAGENT	same-scene margin changes	**90.3**	**2.2**	**84.7**	**72.9**	**92.3**

moves CAPAGENT-8B often sets aside readings whose sign is right (Appendix L, Case 4). Held to SpatialCLI-8B's 2.6 tool calls per question instead of 3.0, CAPAGENT-8B still reaches 69.7; retraining SpatialCLI-8B on trajectories built with the new tools reaches 71.9, but needs new training data for every swap (Table 15). Training only a paired packet MLP in place of Single-output-8B's margin head gives 69.0; pairing the 8B agent itself adds the remaining 1.6 points (Appendix G).

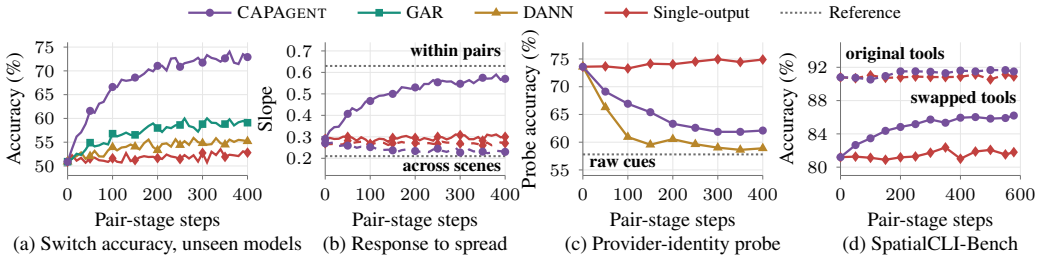
5.3 CONTROLLED MODEL SWAPS

CAPAGENT improves adaptation and retention together (Table 2). Against the single-output margin baseline, switch accuracy rises from 68.4% to 84.7% (52.8% to 72.9% on unseen providers, and 39.3% to 66.1% on outputs of the unseen provider itself, a model that took no part in training), retention from 89.0% to 92.3%, and unsupported commitments fall from 4.1% to 2.2%. Methods trained only on action labels stay below the margin baseline, and DANN and V-REx add only 1.9 and 2.8 switch points. GAR is the strongest baseline on four of the five columns; CAPAGENT is 10.8 switch points and 13.8 unseen points above it. A fixed-radius rule that ignores the cues keeps 94.8% of unchanged actions but reaches only 47.4% Switch, about what ignoring the swap gives (Table 16), so both metrics are needed.

Ground truth at a matched budget. Since the references come from our calibration, we also score actions against ground truth. False proceeds fall from 3.0% to 1.6%, invalid cache reuse from 3.5% to 1.9%, and acting on a wrong or absent object from 6.2% to 3.3% (Table 17). If CAPAGENT may defer no more often than the single-output policy, the three rates are 2.0%, 2.2%, and 3.8%, so most of the reduction comes from better decisions rather than more deferrals. On unseen replacements the agent's own ranges contain the true value 85.2% of the time (nominal 90%), against 72.4% for single-output training (Table 18). Recalibration needs ground-truth 3D scans of new capture units: with 20 units per unseen provider it reaches 71.5% unseen switch accuracy, close to what CAPAGENT reaches without labels, and with 50 units 81.0% (Table 19).

Table 3: **What is paired, and with which loss.** Controlled core (%). None: single-output training on 2 or 6 treatments. GAR loss: GAR's difference and correlation terms. Brackets: 95% intervals.

Pairs	Pair loss	Switch↑	Unseen↑	Retain↑
None (2)	–	68.4 [65.9, 70.8]	52.8	89.0
None (6)	–	74.9 [72.4, 77.3]	–	89.4
All pairs of a batch	GAR (published)	73.9 [71.3, 76.4]	59.1	89.7
Different scenes	CAPAGENT	69.1 [66.6, 71.6]	53.4	87.6
Same scene	GAR loss	78.8 [76.3, 81.2]	64.9	89.8
Same scene	GAR loss + cross + retain	81.8 [79.3, 84.2]	69.2	92.0
Same scene	CAPAGENT	**84.7** [81.9, 87.4]	**72.9**	**92.3**



(a) Switch accuracy, unseen models (b) Response to spread (c) Provider-identity probe (d) SpatialCLI-Bench

Figure 4: **What changes during the pair stage.** (a)–(c) Controlled policy: (a) unseen switch accuracy; (b) radius-on-spread slope within pairs (solid) and across scenes (dashed), dotted: reference; (c) provider probe on the anchor state, dotted: raw cues. (d) SpatialCLI-Bench, swapped (solid) and original (dashed) tools.

5.4 WHAT DRIVES THE GAIN

Matching the scene is what makes pairs useful. With CAPAGENT's loss, pairs from different scenes reach 69.1% switch accuracy, barely above single-output training, as Proposition 1 predicts; same-scene pairs reach 84.7%. GAR's published objective over all pairs of a batch reaches 73.9%, yet its within-pair slope (0.32) barely moves from that of single-output training (0.30; Appendix F). Applying GAR's two pair terms to same-scene pairs instead, with L_{warm} in place of its pointwise MAE, raises switch accuracy to 78.8%. Seeing more outputs does not explain the gain (single-output training on all six treatments reaches 74.9%), and reversing the paired targets drops CAPAGENT to 29.4%, so the direction of the targets matters.

Our pair loss adds to same-scene pairing. On the same pairs, CAPAGENT's loss beats GAR's loss plus our cross and retain terms by 2.9 points [1.2, 4.6], and by 0.8 [0.2, 1.4] in SpatialCLI's loop (70.6 against 69.8); with GAR's terms on the same margins as our shift loss, the gap is still 2.3 [0.7, 3.9]. Each term has a role (Table 20). Without cross, shift stays below GAR's loss (76.5% against 78.8%). Shift and cross together reach the highest switch accuracy (87.9%) but keep only 71.0% of retain decisions; the retain term costs 3.2 switch points and restores retention to 92.3%.

5.5 HOW PAIRING CHANGES THE USE OF QUALITY CUES

Pairs raise the response to a provider change. We regress the predicted radius (the half-width of the depth range) on spread across scenes for a fixed provider, and within same-scene pairs with one intercept per provider pair. The reference slopes are 0.21 and 0.63, the threefold gap of Section 1; the single-output agent gives 0.27 and 0.30. During the pair stage CAPAGENT's within-pair slope rises to 0.57 (Figure 4b), together with unseen switch accuracy and accuracy in the loop (Figure 4a, d). With different-scene pairs the same loss leaves it at 0.31, as Proposition 1 predicts. Pairing barely changes the error in the center of a predicted depth range (0.9 to 0.8 cm on unseen outputs) and cuts the error in its radius from 3.9 to 2.3 cm (Table 18), so the gain comes from judging how far to trust a reading. Within pairs, the correlation of predicted and reference radius changes rises from 0.18 to 0.81.

Table 4: **Cues, and where the size of the change matters.** Controlled core (%). (a) Agents retrained with reduced packets (reference slopes 0.21 / 0.63). (b) Switch pairs in thirds of $\min_j |\phi_{B,j}|$, GAR loss with cross and retain; last column: % meeting Eq. 5 (GAR / CAPAGENT).

(a) Removing cues

	Single	Paired		Paired slopes
Packet	Switch↑	Switch↑	Retain↑	across / within
All cues	68.4	**84.7**	**92.3**	0.23 / 0.57
No edge cue	67.1	78.0	89.1	0.39 / 0.48
Measurement only	51.9	53.2	84.3	–

(b) Switch accuracy by distance to the threshold

Distance	Single	GAR loss	CAPAGENT	Eq. 5 holds
Near	49.6	70.2	**76.7**	29 / 38
Middle	69.8	83.9	**86.1**	57 / 66
Far	85.7	91.2	**91.4**	84 / 88

Pairing also changes how the image cue is used. The across-scene slope moves the other way, from 0.27 to 0.23: across scenes, spread rises with edge energy, so this slope also carries the edge weight. In a regression with provider fixed effects, one standard deviation more edge energy lowers the predicted radius by 0.001 m for the single-output agent, 0.011 m after pairing, and 0.014 m for the references (Appendix B), a rise that Proposition 1 predicts. Retraining without edge energy (Table 4a) moves the paired agent's slopes together (0.39 and 0.48), cuts its switch gain from 16.3 to 10.9 points, and lowers retention to 89.1%; with the measurement alone, pairing gains 1.3 points.

Removing provider information is not enough. Single-output training makes the provider easier to decode from the anchor state (73.6% after warm-up to 74.9%; raw cues give 57.8%), and pairing lowers this to 62.1% (Figure 4c). DANN lowers it further, to 58.9%, yet reaches only 55.2% unseen switch accuracy. Naming the provider helps the single-output agent only on seen providers (unseen: 52.8% to 51.8%; Table 19).

The size of the predicted change matters near a threshold. Far from a threshold the two pair losses are within 0.2 points (Table 4b); in the nearest third CAPAGENT leads by 6.5, and its predictions meet Equation 5 on 38% of pairs (GAR's loss: 29%).

5.6 BREADTH AND LIMITS

On all 49,416 indoor pairs switch accuracy rises from 60.2% to 77.8%, and outdoor and grounding policies gain 15.9 and 12.8 points. Pairing helps every policy class we tried, by 10.5 (packet MLP) to 16.3 points (8B). Across ten domain–provider settings, the gain grows with how much less reliable the replacement is than the reference. Transfer is weakest where a swap changes reliability little or improves it: replacements more reliable than any training provider improve from 43.3% to 59.9% (24.2% to 45.9% on their own outputs), and Depth Anything V2 Large (radius ratio 1.15) reaches 62.2%. On the very noisy Metric3D v2 Small switch accuracy is 78.8%, and most remaining errors predict too narrow a range (Table 24). Transfer also needs the cues to show the error. Scaling UniDepthV2's depth by 1.06 makes every clearance read 6% wide while its cues barely move; on its switch outputs, trained policies stay near the fixed-radius rule (11.9–15.2% against 8.1%), and recalibration on 20 labelled units reaches 64.3% (Table 23).

6 CONCLUSION

When a perception model is replaced, the same measurement can support a different decision. Single-output training teaches an agent what quality cues mean across scenes, which understates what they say when the model changes. CAPAGENT learns from same-scene pairs how its decision margins should move, reads one output at a time, and with unseen tools keeps 96% of its standard-benchmark score, level with SpatialCLI-27B. Matching the scene is what makes the pairs work.

Limitations. Our tasks compare a continuous measurement with a threshold, and training needs two providers per capability on shared scenes plus a small calibration split with ground truth. Transfer is weakest when a swap changes reliability little or improves it, and an error that the quality cues do not show needs information from outside the packet, such as labels from the new model or an independent measurement. **Future work.** Training with a wider spread of providers, including stronger ones, targets the first case; a second, independent measurement could cover the second, and same-scene supervision for tools that return masks or plans would go beyond thresholds.

REPRODUCIBILITY STATEMENT

The appendices give the training procedure (Algorithm 1), all hyperparameters, data splits, provider checkpoints and roles, reference construction, task rules, packet definitions, prompts, the proof of Proposition 1, the agent-loop protocol, and the hardware and run times (eight RTX 4090 GPUs; Appendix C.3). Training, evaluation, and CAPSWAP-BENCH construction code is at `https://anonymous.4open.science/r/CAPAgent-8E4C/`.

AI USE STATEMENT

Large language models assisted with manuscript drafting, figure preparation, and code review. The authors designed the study, ran the experiments, and take responsibility for all content.

REFERENCES

Anastasios N. Angelopoulos and Stephen Bates. Conformal Prediction: A Gentle Introduction. *Foundations and Trends in Machine Learning*, 16(4):494–591, 2023. doi: 10.1561/2200000101. URL https://doi.org/10.1561/2200000101.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant Risk Minimization. arXiv preprint arXiv:1907.02893, 2019. URL https://arxiv.org/abs/1907.02893.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL Technical Report. arXiv preprint arXiv:2511.21631, 2025. URL https://arxiv.org/abs/2511.21631.

Gagan Bansal, Besmira Nushi, Ece Kamar, Daniel S. Weld, Walter S. Lasecki, and Eric Horvitz. Updates in Human-AI Teams: Understanding and Addressing the Performance/Compatibility Tradeoff. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):2429–2437, 2019. doi: 10.1609/aaai.v33i01.33012429. URL https://doi.org/10.1609/aaai.v33i01.33012429.

Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. ZoeDepth: Zero-shot Transfer by Combining Relative and Metric Depth. arXiv preprint arXiv:2302.12288, 2023. URL https://arxiv.org/abs/2302.12288.

Vineet Bhat, Sungsu Kim, Valts Blukis, Greg Heinrich, Prashanth Krishnamurthy, Ramesh Karri, Stan Birchfield, Farshad Khorrami, and Jonathan Tremblay. BOP-ASK: Object-Interaction Reasoning for Vision-Language Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16746–16757, June 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Bhat_BOP-ASK_Object-Interaction_Reasoning_for_Vision-Language_Models_CVPR_2026_paper.html.

Aleksei Bochkovskii, Amaël Delaunoy, Hugo Germain, Marcel Santos, Yichao Zhou, Stephan R. Richter, and Vladlen Koltun. Depth Pro: Sharp Monocular Metric Depth in Less Than a Second. In *International Conference on Learning Representations*, pp. 75602–75637, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/bc8b2058fd96978a4146f18298cb2d39-Paper-Conference.pdf.

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 Dollár, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. SAM 3: Segment Anything with Concepts. In *International Conference on Learning Representations*, pp. 138846–138923, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/e0982cbc81401df3430ee1ff780dc7a2-Paper-Conference.pdf.

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 (CVPR)*, pp. 37109–37120, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Chen_SpaceTools_Tool-Augmented_Spatial_Reasoning_via_Double_Interactive_RL_CVPR_2026_paper.html.

Tianheng Cheng, Lin Song, Yixiao Ge, Wenyu Liu, Xinggang Wang, and Ying Shan. YOLO-World: Real-Time Open-Vocabulary Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16901–16911, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Cheng_YOLO-World_Real-Time_Open-Vocabulary_Object_Detection_CVPR_2024_paper.html.

Shreyansh Daftry, Sam Zeng, J. Andrew Bagnell, and Martial Hebert. Introspective perception: Learning to predict failures in vision systems. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1743–1750. IEEE, 2016. doi: 10.1109/iros.2016.7759279. URL https://doi.org/10.1109/iros.2016.7759279.

Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Niessner. ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5828–5839, 2017. URL https://openaccess.thecvf.com/content_cvpr_2017/html/Dai_ScanNet_Richly-Annotated_3D_CVPR_2017_paper.html.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-Adversarial Training of Neural Networks. *Journal of Machine Learning Research*, 17(59):1–35, 2016. URL http://jmlr.org/papers/v17/15-239.html.

Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hannaneh Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. Evaluating Models' Local Decision Boundaries via Contrast Sets. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1307–1323. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.findings-emnlp.117. URL https://doi.org/10.18653/v1/2020.findings-emnlp.117.

Saurabh Garg, Sivaraman Balakrishnan, Zachary C. Lipton, Behnam Neyshabur, and Hanie Sedghi. Leveraging Unlabeled Data to Predict Out-of-Distribution Performance. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2201.04234.

Yonatan Geifman and Ran El-Yaniv. Selective Classification for Deep Neural Networks. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/4a8423d5e91fda00bb7e46540e2b0cf1-Paper.pdf.

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. doi: 10.1109/cvpr.2012.6248074. URL https://doi.org/10.1109/cvpr.2012.6248074.

Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. doi: 10.1177/0278364913491297. URL https://doi.org/10.1177/0278364913491297.

Tanmay Gupta and Aniruddha Kembhavi. Visual Programming: Compositional Visual Reasoning Without Training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14953–14962, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Gupta_Visual_Programming_Compositional_Visual_Reasoning_Without_Training_CVPR_2023_paper.html.

Nikolaj Hindsbo, Sina Ehsani, and Pragyana Mishra. SCOPE: Real-Time Natural Language Camera Agent at the Edge: A Sim-to-Real Benchmark and Analysis of Open-Source Vision and Language Agents for PTZ Camera Tasks. In *Proceedings of the 21st ACM/IEEE International Conference on Human-Robot Interaction*, pp. 553–561. ACM, 2026. doi: 10.1145/3757279.3785641. URL https://doi.org/10.1145/3757279.3785641.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2106.09685.

Mu Hu, Wei Yin, Chi Zhang, Zhipeng Cai, Xiaoxiao Long, Hao Chen, Kaixuan Wang, Gang Yu, Chunhua Shen, and Shaojie Shen. Metric3D v2: A Versatile Monocular Geometric Foundation Model for Zero-Shot Metric Depth and Surface Normal Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):10579–10596, 2024. doi: 10.1109/tpami.2024.3444912. URL https://doi.org/10.1109/tpami.2024.3444912.

Divyansh Kaushik, Eduard Hovy, and Zachary C. Lipton. Learning the Difference that Makes a Difference with Counterfactually-Augmented Data. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1909.12434.

David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. Out-of-Distribution Generalization via Risk Extrapolation (REx). In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 5815–5826. PMLR, 2021. URL https://proceedings.mlr.press/v139/krueger21a.html.

Justin Lazarow, David Griffiths, Gefen Kohavi, Francisco Crespo, and Afshin Dehghan. Cubify Anything: Scaling Indoor 3D Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22225–22233, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Lazarow_Cubify_Anything_Scaling_Indoor_3D_Object_Detection_CVPR_2025_paper.html.

Vincent Leroy, Yohann Cabon, and Jerome Revaud. Grounding Image Matching in 3D with MASt3R. In *Computer Vision –ECCV 2024*, pp. 71–91. Springer Nature Switzerland, 2024. ISBN 9783031732201. doi: 10.1007/978-3-031-73220-1_5. URL https://doi.org/10.1007/978-3-031-73220-1_5.

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 *International Conference on Learning Representations*, pp. 141261–141285, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/e4cd50120b6d7e8daff1749d6bbaa889-Paper-Conference.pdf.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding DINO: Marrying DINO with Grounded Pre-training for Open-Set Object Detection. In *Computer Vision –ECCV 2024*, pp. 38–55. Springer Nature Switzerland, 2024. ISBN 9783031729706. doi: 10.1007/978-3-031-72970-6_3. URL https://doi.org/10.1007/978-3-031-72970-6_3.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. ToolSandbox: A Stateful, Conversational, Interactive Evaluation Benchmark for LLM Tool Use Capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183. Association for Computational Linguistics, 2025. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.65. URL https://doi.org/10.18653/v1/2025.findings-naacl.65.

Matthias Minderer, Alexey Gritsenko, and Neil Houlsby. Scaling Open-Vocabulary Object Detection. In *Advances in Neural Information Processing Systems*, volume 36, pp. 72983–73007. Curran Associates, Inc., 2023. doi: 10.52202/075280-3191. URL https://doi.org/10.52202/075280-3191.

Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational Knowledge Distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3967–3976, 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/html/Park_Relational_Knowledge_Distillation_CVPR_2019_paper.html.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large Language Model Connected with Massive APIs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 126544–126565. Curran Associates, Inc., 2024. doi: 10.52202/079017-4020. URL https://doi.org/10.52202/079017-4020.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The Berkeley Function Calling Leaderboard (BFCL): From Tool Use to Agentic Evaluation of Large Language Models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 48371–48392. PMLR, 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Luigi Piccinelli, Yung-Hsu Yang, Christos Sakaridis, Mattia Segu, Siyuan Li, Luc Van Gool, and Fisher Yu. UniDepth: Universal Monocular Metric Depth Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10106–10116, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Piccinelli_UniDepth_Universal_Monocular_Metric_Depth_Estimation_CVPR_2024_paper.html.

Luigi Piccinelli, Christos Sakaridis, Yung-Hsu Yang, Mattia Segu, Siyuan Li, Wim Abbeloos, and Luc Van Gool. UniDepthV2: Universal Monocular Metric Depth Estimation Made Simpler. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(3):2354–2367, 2026. doi: 10.1109/tpami.2025.3628473. URL https://doi.org/10.1109/tpami.2025.3628473.

Matteo Poggi, Filippo Aleotti, Fabio Tosi, and Stefano Mattoccia. On the Uncertainty of Self-Supervised Monocular Depth Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3227–3237, 2020. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Poggi_On_the_Uncertainty_of_Self-Supervised_Monocular_Depth_Estimation_CVPR_2020_paper.html.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. In *International Conference on Learning Representations*, pp. 9695–9717, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/28e50ee5b72e90b50e7196fde8ea260e-Paper-Conference.pdf.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. In *Advances in Neural Information Processing Systems*, volume 36, pp. 53728–53741. Curran Associates, Inc., 2023. doi: 10.52202/075280-2338. URL https://doi.org/10.52202/075280-2338.

Allen Z. Ren, Anushri Dixit, Alexandra Bodrova, Sumeet Singh, Stephen Tu, Noah Brown, Peng Xu, Leila Takayama, Fei Xia, Jake Varley, Zhenjia Xu, Dorsa Sadigh, Andy Zeng, and Anirudha

Majumdar. Robots That Ask For Help: Uncertainty Alignment for Large Language Model Planners. In *Proceedings of The 7th Conference on Robot Learning*, volume 229, pp. 661–682. PMLR, 2023. URL https://proceedings.mlr.press/v229/ren23a.html.

Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally Robust Neural Networks for Group Shifts: On the Importance of Regularization for Worst-Case Generalization. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1911.08731.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language Models Can Teach Themselves to Use Tools. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551. Curran Associates, Inc., 2023. doi: 10.52202/075280-2997. URL https://doi.org/10.52202/075280-2997.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv preprint arXiv:2402.03300, 2024. URL https://arxiv.org/abs/2402.03300.

Yantao Shen, Yuanjun Xiong, Wei Xia, and Stefano Soatto. Towards Backward-Compatible Representation Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6368–6377, 2020. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Shen_Towards_Backward-Compatible_Representation_Learning_CVPR_2020_paper.html.

Megha Srivastava, Besmira Nushi, Ece Kamar, Shital Shah, and Eric Horvitz. An Empirical Analysis of Backward Compatibility in Machine Learning Systems. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 3272–3280. ACM, 2020. doi: 10.1145/3394486.3403379. URL https://doi.org/10.1145/3394486.3403379.

Dídac Surís, Sachit Menon, and Carl Vondrick. ViperGPT: Visual Inference via Python Execution for Reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11888–11898, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Suris_ViperGPT_Visual_Inference_via_Python_Execution_for_Reasoning_ICCV_2023_paper.html.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully Test-time Adaptation by Entropy Minimization. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2006.10726.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual Geometry Grounded Transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5294–5306, 2025a. URL https://openaccess.thecvf.com/content/CVPR2025/html/Wang_VGGT_Visual_Geometry_Grounded_Transformer_CVPR_2025_paper.html.

Qineng Wang, Baiqiao Yin, Pingyue Zhang, Jianshu Zhang, Kangrui Wang, Zihan Wang, Jieyu Zhang, Keshigeyan Chandrasegaran, Han Liu, Ranjay Krishna, Saining Xie, Jiajun Wu, Fei-Fei Li, and Manling Li. MindCube: Spatial Mental Modeling from Limited Views. In *International Conference on Learning Representations*, pp. 82559–82622, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/858de961874816f894285ec02e50c7fe-Paper-Conference.pdf.

Ruicheng Wang, Sicheng Xu, Yue Dong, Yu Deng, Jianfeng Xiang, Zelong Lv, Guangzhong Sun, Xin Tong, and Jiaolong Yang. MoGe-2: Accurate Monocular Geometry with Metric Scale and Sharp Details. In *Advances in Neural Information Processing Systems*, volume 38, pp. 35928–35959. Curran Associates, Inc., 2025b. doi: 10.52202/085713-1207. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/336572db3e99930814d6b328d4220cb6-Abstract-Conference.html.

Shihao Wang, Shilong Liu, Yuanguo Kuang, Xinyu Wei, Yangzhou Liu, Zhiqi Li, Yunze Man, Guo Chen, Andrew Tao, Guilin Liu, Jan Kautz, Lei Zhang, and Zhiding Yu. LocateAnything: Fast and High-Quality Vision-Language Grounding with Parallel Box Decoding. arXiv preprint arXiv:2605.27365, 2026b. URL https://arxiv.org/abs/2605.27365.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3D Vision Made Easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20697–20709, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Wang_DUSt3R_Geometric_3D_Vision_Made_Easy_CVPR_2024_paper.html.

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 *International Conference on Learning Representations*, pp. 10481–10497, 2026c. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/11a09e0aaa74867c6b0719c639fc09f8-Paper-Conference.pdf.

Zehan Wang, Ziang Zhang, Jiayang Xu, Jialei Wang, Tianyu Pang, Chao Du, Hengshuang Zhao, and Zhou Zhao. Orient Anything V2: Unifying Orientation and Rotation Understanding. In *Advances in Neural Information Processing Systems*, volume 38, pp. 52622–52645. Curran Associates, Inc., 2025c. doi: 10.52202/085713-1754. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/4bda4364bf10a454769fdbeb8c02df16-Abstract-Conference.html.

Jeffrey M. Wooldridge. *Econometric Analysis of Cross Section and Panel Data*. MIT Press, 2nd edition, 2010. ISBN 9780262232586. URL https://mitpress.mit.edu/9780262232586/econometric-analysis-of-cross-section-and-panel-data/.

Duo Wu, Jinghe Wang, Yuan Meng, Yanning Zhang, Le Sun, and Zhi Wang. CATP-LLM: Empowering Large Language Models for Cost-Aware Tool Planning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8699–8709, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Wu_CATP-LLM_Empowering_Large_Language_Models_for_Cost-Aware_Tool_Planning_ICCV_2025_paper.html.

Sijie Yan, Yuanjun Xiong, Kaustav Kundu, Shuo Yang, Siqi Deng, Meng Wang, Wei Xia, and Stefano Soatto. Positive-Congruent Training: Towards Regression-Free Model Updates. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14299–14308, 2021. URL https://openaccess.thecvf.com/content/CVPR2021/html/Yan_Positive-Congruent_Training_Towards_Regression-Free_Model_Updates_CVPR_2021_paper.html.

Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth Anything V2. In *Advances in Neural Information Processing Systems*, volume 37, pp. 21875–21911. Curran Associates, Inc., 2024. doi: 10.52202/079017-0688. URL https://doi.org/10.52202/079017-0688.

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*, pp. 157051–157088, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/fe5d4bd3e2af823701b5d8f70a3c7602-Paper-Conference.pdf.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

Junjie Ye, Yilong Wu, Songyang Gao, Caishuang Huang, Sixian Li, Guanyu Li, Xiaoran Fan, Qi Zhang, Tao Gui, and Xuanjing Huang. RoTBench: A Multi-Level Benchmark for Evaluating the Robustness of Large Language Models in Tool Learning. In *Proceedings of the*

2024 Conference on Empirical Methods in Natural Language Processing, pp. 313–333. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.19. URL https://doi.org/10.18653/v1/2024.emnlp-main.19.

Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. ScanNet++: A High-Fidelity Dataset of 3D Indoor Scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12–22, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Yeshwanth_ScanNet_A_High-Fidelity_Dataset_of_3D_Indoor_Scenes_ICCV_2023_paper.html.

Yang Zhou, Zixuan Huang, Sunzhu Li, Zhuo Yang, Chen Zhang, Shunian Chen, Caijun Yan, Jianyao Xu, Shunyu Liu, Weijie Fu, Peiliang Li, Xiaozhi Chen, and Yuxiang Cai. SpatialCLI: Learning to Reason With Spatial Tools, Then Without Them. arXiv preprint arXiv:2607.27703, 2026. URL https://arxiv.org/abs/2607.27703.

Dixian Zhu, Tianbao Yang, and Livnat Jerby. Gradient Aligned Regression via Pairwise Losses. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 80259–80281. PMLR, 2025. URL https://proceedings.mlr.press/v267/zhu25x.html.

APPENDIX CONTENTS

A Algorithm Pseudocode **18**

B Derivation of Proposition 1 **18**

C Detailed Experimental Settings **20**
C.1 Detailed Training Settings 20
C.2 Detailed Evaluation Settings 21
C.3 Hardware and Software Environment 23

D CAPSWAP-BENCH **23**
D.1 Scope and Sample Format 23
D.2 Data Provenance and Splits 24
D.3 Provider Pool and Roles 25
D.4 Reference Construction 25
D.5 Pair Construction and Statistics 26
D.6 Human Audit of Queries and Ground Truth 27
D.7 Replay Episodes 27

E Packets, Task Rules, and Tool Interfaces **28**
E.1 Packet Fields and Cue Definitions 28
E.2 Task Rules 28
E.3 Tool Interfaces inside SpatialCLI 29
E.4 Evidence Checks and Scales 30

F Baseline Implementations **31**

G Additional Experiments **32**
G.1 Standard Benchmarks inside SpatialCLI 32
G.2 Full Controlled Results 33
G.3 Ground-Truth Outcomes 33
G.4 The Agent's Own Ranges 34
G.5 Extra Information at Deployment 35
G.6 Parts of the Pair Loss 35
G.7 Results by Replacement Provider 36
G.8 Full Evaluations, Other Tracks, and Replay 37
G.9 A Replacement Whose Errors the Cues Do Not Show 37
G.10 Failure Modes 37
G.11 Scaling with Data, Reliability Change, Retain Weight, and Call Budget 38
G.12 Policy Classes 38
G.13 Cost 39

H Tool Calls Before and After a Swap **39**
H.1 One Call per Tool 39
H.2 Checks from Eight Evaluation Questions 41
H.3 A Complete Episode 42

I Visual Examples **43**
I.1 How a Depth Packet Is Built 44
I.2 Spread Across Scenes and Within Pairs 44
I.3 What the Agent Does with Its Checks 45

J Scene Gallery **45**

K Prompt Templates **47**

L Case Studies **50**
L.1 Case 1: A Doorway with an Unseen Depth Model 50
L.2 Case 2: Two Near-Tied Detector Boxes 51
L.3 Case 3: Camera Position with a Swapped Pose Model 53
L.4 Case 4 (Failure): Under-Trusting a Usable Pose Reading 54

M Limitations and Broader Impact **55**

A ALGORITHM PSEUDOCODE

Algorithm 1 summarizes training and deployment. DECIDE is the only part that runs at deployment; everything above it uses calibration data and pairs of providers that the deployed agent never needs. In the agent loop, DECIDE is called once for every tool return that carries a check, and its three outcomes map to the three things an agent can do with a measurement: use it, query again, or set it aside. q_τ is the rule of the stated check. For a check with one interval, such as a depth order, it commits exactly when all the margins it reads have the same sign; the controlled tasks use the rules of Appendix E.2, which also cover several cached views and the presence and localization margins of grounding.

Algorithm 1 CAPAGENT: training on same-scene pairs, deployment on single outputs

Require: Initial policy π_0 (Qwen3-VL-8B-Instruct with LoRA adapters and margin head); training providers $\mathcal{P}$ per capability; calibration, training, and validation units $\mathcal{X}_{\text{cal}}, \mathcal{X}_{\text{tr}}, \mathcal{X}_{\text{val}}$; loss weights $\eta, \lambda_s, \lambda_x, \lambda_r$; step counts T_1, T_2

Ensure: Frozen policy π with margin head f

Calibrate. For each $P \in \mathcal{P}$, fit the residual regressor on half of $\mathcal{X}_{\text{cal}}$ and the conformal quantile on the other half; this gives $r_P(\cdot)$.
Build targets. For every $x \in \mathcal{X}_{\text{tr}}$, task τ, and $P \in \mathcal{P}$: packet $e = P(x, \tau)$, state $\phi(e)$ from r_P and the task rule, action $q(\phi)$, and a target response.
Build pairs. Group packets by (x, τ); every two providers of the same capability give a pair (e_A, e_B) with $d = \phi_B - \phi_A$, the switch set S, the swapped coordinates J, and the coordinates U whose reference margins stay the same.
for $t = 1, \ldots, T_1$ **do** ▷ warm-up on single outputs
    Sample a batch of packets; minimize $L_{\text{SFT}} + \eta L_{\text{state}}$.
end for
for $t = 1, \ldots, T_2$ **do** ▷ pair stage
    Sample a batch of same-scene pairs, half switch and half retain.
    Run e_A and e_B in separate forward passes; read $\hat{\phi}_A, \hat{\phi}_B$ from the anchor states.
    Minimize $L_{\text{warm}}(A) + L_{\text{warm}}(B) + \lambda_s L_{\text{shift}} + \lambda_x L_{\text{cross}} + \lambda_r L_{\text{retain}}$ (Equations 3 and 4).
end for
Select the checkpoint on $\mathcal{X}_{\text{val}}$ (training providers only); freeze π and f.

function DECIDE(e, τ, *check*) ▷ deployment: one packet, no provider name
    check fixes the measurement p, threshold θ, and scale s_τ.
    $\hat{\phi} \leftarrow f(\mathbf{h}_{\text{anchor}}(\pi, e, \tau, \textit{check}))$
    $a \leftarrow q_\tau(\hat{\phi})$ ▷ task rule of the check
    if a commits **then**
        return USE with the relation or its negation
    else
        return RE-QUERY if the call budget allows, otherwise UNUSABLE
    end if
end function

B DERIVATION OF PROPOSITION 1

Model and assumptions. For scene x and provider P, the cue and the error radius are

$$z_P(x) = s(x) + b_P(x), \qquad r_P(x) = \alpha\, s(x) + \beta\, b_P(x) + \epsilon_P(x), \qquad b_P(x) = \mu_P + w_P(x). \tag{6}$$

The scene term s is shared by all providers; μ_P is the provider's average contribution and w_P its variation across scenes. The image cue $u(x)$ is also shared, and $s = tu + v$ is the population linear projection of s on u, so v is uncorrelated with u. We assume that u, v, the w_P, and the ϵ_P have zero mean and are mutually independent, with $\mathrm{Var}(u) = \sigma_u^2$, $\mathrm{Var}(v) = \sigma_v^2$, $\mathrm{Var}(w_P) = \sigma_w^2 > 0$ for every provider, and $\mathrm{Var}(\epsilon_P) = \sigma_\epsilon^2$. Radii here are fluctuations around a positive mean; a shared constant can be absorbed into the offsets. The predictor is $\hat{r}_P = k z_P + h u + c_P$, with one offset c_P per training provider.

Residuals. Substituting the model,

$$\hat{r}_P - r_P = (kt + h - \alpha t)\, u + (k - \alpha)\, v + (k - \beta)\, w_P + \big[c_P + (k - \beta)\mu_P\big] - \epsilon_P. \tag{7}$$

Within a same-scene pair (A, B) the u and v terms are identical and cancel in the difference:

$$(\hat{r}_B - \hat{r}_A) - (r_B - r_A) = (k-\beta)(w_B - w_A) + \big[c_B - c_A + (k-\beta)(\mu_B - \mu_A)\big] - (\epsilon_B - \epsilon_A). \quad (8)$$

Joint objective. The objective of the proposition is $\mathbb{E}\big[(\hat{r}_A - r_A)^2 + (\hat{r}_B - r_B)^2 + \lambda\,((\hat{r}_B - \hat{r}_A) - (r_B - r_A))^2\big]$. Using independence, Equations 7 and 8 give

$$\begin{aligned}\mathcal{J}(k,h,c) = {} & 2(kt + h - \alpha t)^2\sigma_u^2 + 2(k-\alpha)^2\sigma_v^2 + 2(1+\lambda)(k-\beta)^2\sigma_w^2 \\ & + \textstyle\sum_{P\in\{A,B\}} \big(c_P + (k-\beta)\mu_P\big)^2 + \lambda\big(c_B - c_A + (k-\beta)(\mu_B - \mu_A)\big)^2 + \text{const.}\end{aligned} \quad (9)$$

The offsets $c_P = (\beta - k)\mu_P$ set both offset terms to zero, and $h = (\alpha - k)t$ sets the first term to zero. What remains is $2\sigma_v^2(k-\alpha)^2 + 2(1+\lambda)\sigma_w^2(k-\beta)^2$, whose minimizer is

$$k(\lambda) = \frac{\alpha\,\sigma_v^2 + (1+\lambda)\,\beta\,\sigma_w^2}{\sigma_v^2 + (1+\lambda)\sigma_w^2} = (1 - w_\lambda)\,\alpha + w_\lambda\,\beta, \quad (10)$$

with w_λ as in Equation 2. Because w_λ increases with λ, stronger pair supervision moves k monotonically from $k(0)$ toward β. Here σ_v^2 is the scene variation that the image cue leaves unexplained, not the total scene variation. When it exceeds the provider variation, $\sigma_v^2 > \sigma_w^2$ and $w_0 < 1/2$, so the single-output slope $k(0)$ lies closer to α than to β. The optimal image weight $h = (\alpha - k)t$ is zero at $k = \alpha$ and grows in size as k moves toward β: the higher the slope on the cue, the more the image cue is worth reading.

The two responses. Within a pair, the predicted change is $\hat{r}_B - \hat{r}_A = k\,(z_B - z_A) + (c_B - c_A)$, so a regression of the predicted change on the cue change with a provider-pair intercept has slope k. Across scenes for a fixed provider,

$$\frac{\mathrm{Cov}(\hat{r}_P, z_P)}{\mathrm{Var}(z_P)} = k + \frac{h\,t\,\sigma_u^2}{\mathrm{Var}(z_P)} = k + (\alpha - k)\,\rho, \qquad \rho = \frac{t^2\sigma_u^2}{t^2\sigma_u^2 + \sigma_v^2 + \sigma_w^2} \in [0,1]. \quad (11)$$

The reference radius has within-pair slope β and across-scene slope $[\alpha(t^2\sigma_u^2 + \sigma_v^2) + \beta\sigma_w^2]/\mathrm{Var}(z_P)$. With $k = \beta$ the learned across-scene slope differs from the reference one by $(\beta - \alpha)\sigma_v^2/\mathrm{Var}(z_P)$, which is small when the image cue explains most of the scene term. Without an image cue ($t = 0$, $\rho = 0$) both learned slopes equal k, and no choice of k matches both reference slopes unless $\alpha = \beta$. This is the prediction tested by removing edge energy in Table 4a.

The across-scene response. For fixed data, a change of λ moves the within-pair response by Δk and the across-scene response by $(1 - \rho)\Delta k$, in the same direction. The gap between the two responses is $\rho(k - \alpha)$, which grows with k. A fall of the across-scene response therefore comes from a change in how much the agent uses the image cue. If an agent uses a fraction γ of the optimal image weight, $h = \gamma(\alpha - k)t$, its across-scene response is $a = k + \gamma\rho(\alpha - k)$, and

$$\mathrm{d}a = (1 - \gamma\rho)\,\mathrm{d}k - \rho\,(k - \alpha)\,\mathrm{d}\gamma. \quad (12)$$

With $k > \alpha$, the across-scene slope can fall while k rises only if γ grows enough, $\rho(k - \alpha)\,\mathrm{d}\gamma > (1 - \gamma\rho)\,\mathrm{d}k$. A larger γ is thus needed for the fall but does not guarantee it. Section 5.5 observes the fall, and the regression below measures γ directly.

We measure the image weight of a trained network by least squares. Over the depth outputs of the controlled core, we regress the predicted radius on spread, edge energy, and valid fraction with one intercept per provider, and report the coefficient on edge energy times the pooled standard deviation of edge energy. The same regression on the reference radius gives the reference weight. One standard deviation more edge energy lowers the radius by 0.001 m $[-0.001, 0.003]$ for the single-output margin policy, 0.011 m $[0.009, 0.013]$ for CAPAGENT, and 0.014 m $[0.012, 0.016]$ for the reference (scene bootstrap). In the model, this regression on the reference radius recovers slope $k(0)$ and weight $(\alpha - k(0))t$, the single-output optimum, so the reference weight is what a single-output learner could reach. The trained single-output policy uses far less ($\gamma \approx 0.1$), and pairing raises its weight close to that value.

Pairs from different scenes. If A and B come from independent scenes x and x', the u and v terms no longer cancel. The difference loss then adds $\lambda\,[2(kt+h-\alpha t)^2\sigma_u^2+2(k-\alpha)^2\sigma_v^2+2(k-\beta)^2\sigma_w^2]$ plus offset terms, which is λ times the single-output objective. The minimizer is $k(0)$ for every λ: such pairs add a scaled copy of the single-output loss and no information about β.

Pairwise losses over a batch. With squared error, a difference loss over all pairs of a batch equals, up to a constant factor, the variance of the pointwise errors in the batch (Zhu et al., 2025, Theorem 1). The identity is exact for any batch. For outputs drawn independently across scenes, the expected variance is the single-output loss with a free intercept, and its minimizer is $k(0)$. With the optimal offsets and image weight, the error of an output is $e_{iP} = (k-\alpha)v_i + (k-\beta)w_{iP}$ up to noise. If a batch holds m scenes with two provider outputs each,

$$\mathbb{E}\,\widehat{\mathrm{Var}}(e) = \Big(1-\frac{1}{m}\Big)(k-\alpha)^2\sigma_v^2 + \Big(1-\frac{1}{2m}\Big)(k-\beta)^2\sigma_w^2 + \text{const}, \tag{13}$$

so the provider term gets slightly more weight and the minimizer moves from $k(0)$ toward β by an amount of order $1/m$. Explicit same-scene pairs move the weight on β from w_0 to w_λ, a change that does not shrink with the batch size. GAR follows the single-output schedule, so its pair-stage batches hold 16 same-scene pairs ($m = 16$), and 480 of the 496 pairs in a batch change the scene. GAR's measured within-pair slope is 0.32 (Appendix F).

Offsets and unseen providers. The optimal offsets $c_P = (\beta-k)\mu_P$ carry the part of each training provider's average reliability that the slope misses. At $k(0)$ they are large, and a predictor that can recognize the provider can use them; at $k = \beta$ they vanish. For an unseen provider P' the predictor has no matching offset and uses some $\bar{c}$. The error of its predicted change relative to provider A on the same scene is

$$(k-\beta)\,(b_{P'}-b_A) + (\bar{c}-c_A) - (\epsilon_{P'}-\epsilon_A), \qquad k(0)-\beta = \frac{(\alpha-\beta)\,\sigma_v^2}{\sigma_v^2+\sigma_w^2}. \tag{14}$$

The first term grows with how different the new provider is and shrinks as k approaches β. This is the linear version of two observations: single-output training makes provider identity easier to decode (Section 5.5), and the gain from pairing grows with the reliability gap of the replacement (Section 5.6 and Figure 5b).

Scene–provider interactions. The additive model makes the scene term cancel exactly within a pair. Pairing also supervises interactions between scene and provider, such as depth models that are noisier on reflective surfaces, which the closed form leaves out.

What carries to a new provider. The model gives every provider the same α and β, and this shared link is what lets a slope learned on training providers apply to a new one. Suppose a new provider P' has an extra error term $\kappa_{P'}(x)$ that its cue does not reflect, $r_{P'} = \alpha s+\beta b_{P'}+\kappa_{P'}+\epsilon_{P'}$, for example a scale bias that leaves the spread of its depth unchanged. Every predictor of the form $kz+hu+c$ then misses $\kappa_{P'}$ entirely, whatever k is, because nothing it reads depends on $\kappa_{P'}$. Pairing moves k toward β and so fixes the part of the error that the cue shows. The part it does not show needs information from outside the packet, such as a few labelled scenes of P' or an independent measurement of the same quantity. Table 23 builds such a provider and measures both parts.

From state error to an action. Since $\hat{\phi}_B - \phi_B = (\hat{\phi}_A - \phi_A) + (\hat{d} - d)$, the triangle inequality gives $|\hat{\phi}_{B,j} - \phi_{B,j}| \le |\hat{\phi}_{A,j} - \phi_{A,j}| + |\hat{d}_j - d_j|$. If the right side is smaller than $|\phi_{B,j}|$, then $\hat{\phi}_{B,j}$ has the sign of $\phi_{B,j}$, which is Equation 5. When this holds for every coordinate that q reads, the predicted action equals the reference action. Errors in the two terms can also cancel, so a decision near a threshold can be right when the condition fails (Table 4b).

C Detailed Experimental Settings

C.1 Detailed Training Settings

Initial model. All language-model policies start from Qwen3-VL-8B-Instruct (Bai et al., 2025), the base model of SpatialCLI-8B (Zhou et al., 2026). We keep the vision encoder and all language

weights frozen and train LoRA adapters (Hu et al., 2022) on the query and value projections of every layer of its Qwen3-8B language model, together with the margin head. Table 25 also reports a policy trained from the SpatialCLI-8B checkpoint.

Training data for the indoor policy. The 200 training capture units give 1,784 depth queries and 1,640 pose queries. The two counts differ because some scans contain more usable openings than usable cached views. Each query is assigned one of the three task limits uniformly at random, so the limits are equally represented without tripling the number of near-duplicate packets. Every query is run through the two training providers of its capability (UniDepth and ZoeDepth-NK for depth, VGGT and DUSt3R for pose). This gives 6,848 single packets and 3,424 same-scene pairs, of which 1,003 (29.3%) are switch pairs. A record always shows the agent one depth query and one pose query of the same unit, which is why the response has one anchor token per capability: a depth query is shown with a pose query of its unit, taken in turn, and a pose query with a depth query. Within a pair only the provider of the queried capability changes. The other query keeps the reference provider's packet, so its reference margins stay the same and form U; under a depth swap the pose query still sees a new `range_m`, because that field comes from the depth packet.

Warm-up reads the 6,848 packets once: 214 steps with 32 packets per step. The pair stage runs 398 steps with 16 pairs per step, about 1.9 passes over the 3,424 pairs. Each pair-stage batch holds 8 switch pairs and 8 retain pairs. We balance the batch because switch pairs are the minority and carry all of the cross loss; the retain term acts on the untouched query of every pair, switch or retain. The single-output control reads the same packets for the same 612 steps, with the pair losses removed, so both policies see the same data and use the same compute. The None (6) row of Table 3 is this control with the two indoor training treatments replaced by all six core treatments on the same 200 units.

Other policies. The outdoor policy trains on 200 KITTI frames with four distance queries each, run through UniDepth and ZoeDepth-NK (1,600 packets, 800 pairs). The grounding policy trains on 200 ScanNet++ queries run through Grounding DINO Swin-B and Swin-T (400 packets, 200 pairs). Both use the schedule of Table 5, with the number of steps set to one pass for warm-up and two passes for the pair stage.

Adapters used inside SpatialCLI. The adapters for the agent-loop experiments first learn SpatialCLI's tool-calling format: a tool-format stage reads SpatialCLI's released cold-start SFT trajectories once, with the LoRA settings of Table 5 and up to 8,192 tokens per sequence. All four trained agents of Table 1 continue from this one checkpoint. They then train on the union of the indoor and grounding training sets plus 2,000 check-format examples. The check-format examples are built from training scenes with training providers only: 800 depth-ordering checks, 500 signed camera-motion checks, 400 object-presence checks, and 300 localization checks. Their targets come from the scene annotations through the task rules of Appendix E.4. They come as 1,000 same-scene pairs, so they also enter the pair stage. This run uses 289 warm-up steps and 578 pair-stage steps with the batch sizes above. Check-format examples carry SpatialCLI's system prompt and the image, so this run allows 4,096 tokens per sequence. Tool-use SFT-8B, Single-output-8B, GAR-8B, and CAPAGENT-8B in Table 1 all train on exactly this data. No benchmark question, benchmark image, or replacement provider is used.

Checkpoint selection. We evaluate on the validation units every 50 steps and keep the checkpoint with the highest harmonic mean of switch accuracy and untouched-capability accuracy. All margin-head policies, including the single-output control and the GAR variants, use this rule, and validation uses only training providers.

C.2 Detailed Evaluation Settings

Scoring on CapSwap-Bench. For every policy with a margin head, the action is read from the predicted state through the fixed rule q of Appendix E.2; the generated explanation is not parsed. For policies that answer in text (trajectory SFT, SFT + GRPO, paired-input SFT, and DPO without a head), we parse the action token that follows the `ACTION:` tag. A missing or malformed action counts as wrong, and it does not count as a commitment in Unsup. Parse failures affect 0.3% of trajectory-SFT outputs.

Table 5: **Training configuration** of the indoor policy. The outdoor and grounding policies change only the number of steps; the adapters used inside SpatialCLI also allow 4,096 tokens per sequence (see text).

Category	Warm-up (single outputs)	Pair stage (same-scene pairs)
Objective	$L_{\mathrm{SFT}} + \eta L_{\mathrm{state}}$, $\eta = 1.0$	$L_{\mathrm{warm}}(A) + L_{\mathrm{warm}}(B) + L_{\mathrm{pair}}$
Pair loss weights	–	$\lambda_s = 1.0$, $\lambda_x = 0.5$, $\lambda_r = 0.25$
Margin loss	Smooth-L_1, transition 0.05 (margin units)	same
Soft rule in L_{cross}	–	sigmoid, temperature $T = 0.1$
Optimizer	AdamW, $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$	same
Learning rate	1×10^{-4} (LoRA), 5×10^{-4} (margin head)	same
Schedule	5% linear warm-up, cosine decay to 0	restarted, same shape
Weight decay / gradient clip	0.01 / 1.0	same
Precision	bf16	bf16
Batch (8 GPUs)	32 packets	16 pairs (8 switch, 8 retain)
Per GPU	2 packets $\times$ 2 accumulation steps	1 pair $\times$ 2 accumulation steps
Steps	214 (one pass)	398 (about 1.9 passes)
Maximum sequence length	1,024 tokens	1,024 tokens per packet
Memory	RTX 4090 (24 GB); frozen weights sharded with FSDP; gradient checkpointing	
LoRA	rank 16, scaling 32, dropout 0.05, on `q_proj` and `v_proj` of all 36 language layers	
Margin head	shared two-layer trunk (width 1,024, GELU, LayerNorm) and one linear projection per capability	
Head outputs	a center and a softplus half-width per range: depth 1 range, pose 6 (3 cache entries $\times$ 2 components), grounding 1 presence range and 1 one-sided localization margin	
Trainable parameters	12.6M in total (adapters and margin head)	
Checkpoint selection	every 50 steps, harmonic mean of switch and untouched accuracy on validation	

Acc averages over all scored outputs. Switch and Retain average over the two outputs of each switch or retain pair, so each pair contributes two decisions. Unseen restricts Switch to pairs whose replacement provider was not used in training. Unsup. counts outputs whose action is `PROCEED`, `USE-CACHE`, or `ACT` while the reference action is a different one, divided by all outputs. On the controlled core, Acc equals $0.2667\,\mathrm{Switch} + 0.7333\,\mathrm{Retain}$, since 816 of the 3,060 pairs are switch pairs.

Standard benchmarks. We follow SpatialCLI's evaluation code and parsers (Zhou et al., 2026). SpatialCLI-Bench has 516 six-choice questions. MindCube (Wang et al., 2026a) is scored by multiple-choice accuracy. From MMSI-Bench (Yang et al., 2026) we use the Motion-Cam and Pos-Cam-Cam subsets. DA-2K (Yang et al., 2024) asks which of two marked points is closer. BOPASK (Bhat et al., 2026) Trajectory and Object-Rearrangement are scored with SpatialCLI's point-set score: with predicted and reference point sets $\hat{P}$ and P^* on a $[0, 1]$ image scale,

$$d(\hat{P}, P^*) = \tfrac{1}{2}\Big(\tfrac{1}{|\hat{P}|}\sum_{\hat{p}\in\hat{P}}\min_{p\in P^*}\|\hat{p}-p\|_2 + \tfrac{1}{|P^*|}\sum_{p\in P^*}\min_{\hat{p}\in\hat{P}}\|p-\hat{p}\|_2\Big), \qquad s = \max\big(0,\, 1 - d/0.15\big), \tag{15}$$

and an output without valid points scores zero. Avg is SpatialCLI's macro average: the two MMSI subsets are averaged first, the two BOPASK subsets are averaged first, and the five benchmark scores are then averaged with equal weight. Each reported score is the mean of three evaluation runs. The frontier models run through their public APIs with SpatialCLI's prompt and the same four tools; their original-tool numbers are those reported by SpatialCLI. Our reruns of SpatialCLI-8B with the original tools are within 0.6 points of its reported scores on every benchmark; for the frontier models we reran MindCube and DA-2K with Qwen3.5-397B-A17B, which also differ by at most 0.6 points.

Intervals and paired differences. All intervals are 95% percentile intervals from 10,000 bootstrap resamples. On CAPSWAP-BENCH we resample physical spaces: a space enters or leaves a resample with all its scans, so rescans of one room and every pair built from them move together. We call this the scene bootstrap; every CAPSWAP-BENCH interval in the paper uses it. On the outdoor track we resample drives. On the standard benchmarks we resample questions, except for MindCube, where several questions share one image set and we resample image sets. For a difference between two methods we use the same resampled indices for both methods and report the interval of the difference, which is narrower than comparing two separate intervals because both methods face the same hard scenes.

Table 6: **Evaluation configuration.** The SpatialCLI settings are copied from its release so that our SpatialCLI-8B runs reproduce its reported scores.

Category	Configuration
CAPSWAP-BENCH decisions	margin head + fixed rule; one packet per forward pass; no provider name
CAPSWAP-BENCH explanations	greedy decoding, at most 256 new tokens (not scored)
Text-action baselines	greedy decoding; action parsed after `ACTION:`
SpatialCLI sampling	temperature 1.0, top-p 0.95, top-k 20, min-p 0.0
SpatialCLI penalties	presence penalty 1.5, repetition penalty 1.0
SpatialCLI limits	40,960 generated tokens, at most 10 tool calls per question
SpatialCLI runs	mean of three evaluation runs
Call-capped CAPAGENT-8B	per-question cap = calls made by SpatialCLI-8B (no labels)
Retrained SpatialCLI-8B	SpatialCLI's SFT + RL recipe, data regenerated with the new tools
Intervals	10,000 bootstrap resamples, 95% percentile
Resampling unit	physical space with all its scans (CAPSWAP-BENCH), drive (KITTI), question or image set (benchmarks)

C.3 HARDWARE AND SOFTWARE ENVIRONMENT

All training and evaluation runs used one server with eight NVIDIA GeForce RTX 4090 GPUs (24 GB each), connected over PCIe 4.0 without NVLink, and 512 GB of host memory. GPT-5.6 Sol and Qwen3.5-397B-A17B are the only models we did not run on this server; they are called through their APIs.

Training on 24 GB cards. The frozen bf16 weights of an 8B policy take about 17.5 GB. A full copy on every card leaves too little memory for a pair of 1,024-token packets, so we shard the frozen weights across the eight cards with FSDP and use gradient checkpointing. Each card takes two packets per micro-batch (one pair in the pair stage) and accumulates gradients over two micro-batches, which gives the batches of 32 packets and 16 pairs in Table 5. Losses defined over a whole batch, GAR's terms and the V-REx penalty, use all 32 outputs: each card runs both forward passes, the head outputs are gathered across the cards, and only then does the backward pass start. Most of a training step is spent gathering the sharded weights over PCIe. Training one 8B policy takes 3.8 hours on the eight cards (30.5 GPU-hours) with the indoor schedule of Table 5, and 5.4 hours (43.2 GPU-hours) for the 867 steps of the adapters used inside SpatialCLI; both times include validation every 50 steps. The shared tool-format stage before them runs once and takes 27 GPU-hours. The retrained SpatialCLI-8B of Table 15 is our largest run. It uses SpatialCLI's verl setup with FSDP parameters and optimizer states offloaded to host memory and with vLLM rollouts on pairs of cards, and its SFT and RL stages take about 360 GPU-hours.

Perception providers. Each provider runs on a single card. The pose models see at most four frames per query (the query frame and up to three cached frames), so every provider fits in 24 GB. Running all 15 providers over the calibration, training, validation, and evaluation pools once took about 650 GPU-hours, and we cache every provider output so that all policies read identical packets.

Agent loop. Inside SpatialCLI's loop an 8B agent is served by vLLM with tensor parallelism 2: after 17.5 GB of weights, one card has too little memory left for the key-value cache of a 40,960-token generation. Three such replicas use six cards. The seventh card hosts the perception tools, and the eighth holds a copy of the adapted 8B model that computes the anchor states for the margin head. SpatialCLI-27B, which we only evaluate, runs from its released checkpoint with tensor parallelism across all eight cards; vLLM is limited to 80% of each card's memory so that the tools fit beside it.

Software. The software stack was Ubuntu 22.04.5 LTS, CUDA 12.8, Python 3.12.3, PyTorch 2.9.0 (with FSDP), Transformers 5.5.4, PEFT 0.18.0, vLLM 0.18.0 for agent-loop inference, verl for the retrained SpatialCLI-8B, LightGBM 4.6.0 for the residual regressors, OpenCV 4.11.0 for image cues and SIFT matching, and Open3D 0.19.0 for mesh-based ground truth.

D CAPSWAP-BENCH

D.1 SCOPE AND SAMPLE FORMAT

CAPSWAP-BENCH tests one question: when the perception model behind a tool is replaced, does a frozen agent still take the right action? Each record is a pair of packets for the same scene, the same

query, and the same task, produced by two providers. The agent never sees the provider names, the reference radii, or the ground truth; these fields are stored under `_eval` and used only for scoring. Box D.1 shows the doorway pair of Figure 2; the same scene and query start replay episode 214, Case 1 of Appendix L. For clearance queries, the depth tool reports the clearance it computes from its own depth map in the `depth_m` field, so all depth queries share one field name.

Box D.1: One CapSwap-Bench pair record (evaluation core, passage task)

```
{
  "pair_id": "in-core-0417-d",
  "track": "indoor", "split": "eval_core",
  "unit": "in-eval-0417", "frame": 1260,
  "task": "passage",
  "query": {"type": "clearance", "region_2d": [412, 188, 596, 671],
            "threshold_m": 1.15, "scale_m": 1.15, "task_limit": 0.05},
  "untouched": {"task": "cache_reuse", "query_id": "in-eval-0417-c2",
                "packet": "reference pose provider, shared by A and B"},
  "packets": {
    "A": {"depth_m": 1.19, "spread_m": 0.04, "valid_frac": 0.97, "edge": 0.18},
    "B": {"depth_m": 1.19, "spread_m": 0.11, "valid_frac": 0.91, "edge": 0.18}
  },
  "_eval": {
    "providers": ["unidepth_v1_vitl14", "metric3d_v2_vit_small"],
    "radius_m": [0.03, 0.08],
    "margins": [[0.009, 0.061], [-0.035, 0.104]],
    "reference_actions": ["PROCEED", "RE-MEASURE"],
    "pair_type": "switch",
    "gt_clearance_m": 1.21, "gt_decision": "PROCEED"
  }
}
```

The `edge` value is identical in the two packets because it is computed from the image, not from the depth map (Appendix E). The ground-truth clearance, 1.21 m, lies inside both intervals: output A may commit, while output B does not carry enough evidence to commit, although committing would have been correct in this scene. This is the kind of case where reference agreement and ground-truth outcome differ, and it is why we report both.

D.2 Data Provenance and Splits

Table 7 lists the sources and Table 8 the splits. ScanNet contains 1,513 scans of 707 spaces; all rescans of one space stay in the same split, so the evaluation never sees a room that appeared in training under a different scan. KITTI frames from one drive stay in one split for the same reason. Grounding queries come from the ScanNet++ scenes of the indoor pool and follow the indoor split, so a scene used for training never provides a grounding evaluation query.

Table 7: **Data sources.** Ground truth is used to calibrate the reference radii and to score outcomes; it is never shown to the agent.

Track	Source	Pool	Ground truth
Indoor	ScanNet v2 (Dai et al., 2017)	1,513 scans (707 spaces)	reconstructed mesh, camera poses
Indoor	ScanNet++ v2 (Yeshwanth et al., 2023)	906 scenes	laser-scan mesh, refined poses
Outdoor	KITTI Eigen split (Geiger et al., 2013)	1,037 frames used	LiDAR depth in the image, capped at 80 m
Grounding	ScanNet++ v2 (indoor scenes)	3,980 queries	instance masks rendered into the image

Table 8: **Splits and pair counts.** A capture unit is one indoor scan. The core is a fixed subset of the evaluation split with one query per capability per unit, used for all controlled comparisons; the full evaluations use every query.

Track	Unit	Calib.	Train	Val.	Eval.	Core	Core pairs	Eval. pairs
Indoor (depth, pose)	capture unit	100	200	60	2,059	510	3,060	49,416
Outdoor (depth)	frame	100	200	40	697	300	1,200	2,788
Grounding	query	100	200	50	3,630	350	1,050	10,890
Replay (depth)	scene	–	–	–	120	–	–	480 episodes

The counts follow from the pairing scheme. Each indoor evaluation unit has four depth queries and four pose queries; the four depth treatments are paired with the reference on every depth query and the two pose treatments on every pose query, which gives $4 \times 4 + 2 \times 4 = 24$ pairs per unit and $2{,}059 \times 24 = 49{,}416$ pairs. The core keeps one depth query and one pose query per unit, so it has $510 \times 6 = 3{,}060$ pairs. Each outdoor evaluation frame has one distance query paired with four depth treatments (697×4), and each grounding query is paired with three treatments ($3{,}630 \times 3$). Each grounding scene contributes four queries whose target is present and one whose target is absent.

D.3 Provider Pool and Roles

Table 9 lists the 15 providers, the checkpoints we use, and their role in each policy. Every provider is run at its default input resolution with its official preprocessing. The radius ratio is the mean calibrated radius on the evaluation split divided by that of the track's reference, so it measures how much less (or more) reliable a provider is under our calibration, not its benchmark accuracy.

Table 9: **Providers and their roles.** Ratio: mean calibrated radius relative to the reference of the same track (indoor / outdoor). Ref.: reference; train: training treatment; unseen: held out from training and validation; reliable: held-out provider more reliable than the reference. Depth Pro, used only in the SpatialCLI tool swaps, is not part of CapSwap-Bench.

Provider	Checkpoint	Input	Ratio	Indoor	Outdoor	Grounding
Depth						
UniDepth (Piccinelli et al., 2024)	`unidepth-v1-vitl14`	462×616	1.00 / 1.00	ref.	ref.	–
ZoeDepth-NK (Bhat et al., 2023)	`ZoeD_M12_NK`	384×512	2.62 / 1.38	train	train	–
Depth Anything V2 Large (Yang et al., 2024)	`metric-hypersim / vkitti-vitl`	518 short side	1.15 / 1.92	unseen	unseen	–
Metric3D v2 Large (Hu et al., 2024)	`metric_depth_vit_large_800k`	616×1064	1.52 / 1.24	unseen	unseen	–
Metric3D v2 Small (Hu et al., 2024)	`metric_depth_vit_small_800k`	616×1064	3.41 / 2.05	unseen	unseen	–
UniDepthV2 (Piccinelli et al., 2026)	`unidepth-v2-vitl14`	462×616	0.82 / –	reliable	–	–
MoGe-2 (Wang et al., 2025b)	`moge-2-vitl`	518 short side	0.90 / –	reliable	–	–
Pose						
VGGT (Wang et al., 2025a)	`VGGT-1B`	518	1.00	ref.	–	–
DUSt3R (Wang et al., 2024)	`ViTLarge_BaseDecoder_512_dpt`	512	1.87	train	–	–
MASt3R (Leroy et al., 2024)	`ViTLarge_BaseDecoder_512_catmlpdpt_metric`	512	1.31	unseen	–	–
π^3 (Wang et al., 2026c)	`Pi3`	518	0.88	reliable	–	–
Grounding						
Grounding DINO Swin-B (Liu et al., 2024)	`groundingdino_swinb_cogcoor`	800 short side	1.00	–	–	ref.
Grounding DINO Swin-T (Liu et al., 2024)	`groundingdino_swint_ogc`	800 short side	1.31	–	–	train
OWLv2 (Minderer et al., 2023)	`owlv2-base-patch16-ensemble`	960×960	1.70	–	–	unseen
YOLO-World-L (Cheng et al., 2024)	`yolov8l-worldv2`	640×640	2.12	–	–	unseen

For Depth Anything V2 we use the indoor (Hypersim) metric checkpoint on indoor scenes and the outdoor (Virtual KITTI) checkpoint on KITTI, which is how its authors release metric depth. This explains why its outdoor ratio (1.92) is much larger than its indoor ratio (1.15): the outdoor checkpoint transfers less well to real KITTI frames than the indoor one does to ScanNet. Pose providers that predict geometry up to scale (DUSt3R, VGGT, π^3) are brought to metric scale with one fixed source, the median ratio between the provider's depth and the reference depth map of the query view. Using the same scale source for every pose provider means that a pose swap changes only the pose model.

D.4 Reference Construction

The reference radius of a provider is built from its ground-truth errors on the 100 calibration units, in two steps. Both steps are fitted separately for every provider, because each model has its own error pattern and a pooled fit would blur exactly the differences the benchmark is about.

Residual regressor. A LightGBM quantile regressor predicts the 90th percentile of the absolute error of the provider's measurement (clearance, translation, rotation, presence score, or box IoU). Its features describe the provider's full output around the query and the image: for depth, the mean, median, 5th and 95th percentiles, and robust spread of the provider's depth inside the query region and in a ring around it, the fraction of valid pixels, the depth-gradient magnitude, the number of depth discontinuities larger than 10% of the local depth, and the provider's own confidence map where one exists; for the image, edge energy, local contrast, the fraction of saturated pixels, and texture entropy; for the query, its size and distance to the image border. Pose and grounding use analogous features (correspondence statistics and confidence maps for pose, score distributions and box geometry for grounding). The regressor uses learning rate 0.05 and 0.8 feature and bagging fractions. It is fitted on 50 of the calibration units, and the tree size follows the number of rows

each track provides (Table 10). An indoor unit contributes every query that passes the filters of Appendix D.5, not only the four used in evaluation. An outdoor calibration frame contributes 16 distance queries. A grounding row is one candidate box of a query, with its IoU to the nearest target instance as the label, because the localization error has to be learned for boxes of every rank; the conformal step then uses the reported top box of each of the other 50 queries.

Table 10: **Rows and tree sizes of the residual regressors.** Fit: rows of the 50 fitting units; conformal: scores of the other 50. A grounding fit row is one candidate box; its conformal rows are the reported top boxes. The per-provider isotonic presence map uses the same candidate boxes as the fit.

Track	Calibration unit	Fit rows	Conformal rows	Trees	Leaves	Min. rows per leaf
Indoor depth	capture unit	1,742	1,706	400	31	40
Indoor pose	capture unit	2,436	2,388	400	31	40
Outdoor depth	frame (16 queries)	800	800	200	15	20
Grounding	query (candidate boxes)	486	50	200	15	20

Conformal correction. On the other 50 calibration units we compute the score $|y - p| - \hat{q}(x)$ for every query and set the radius to $r(x) = \hat{q}(x) + \hat{Q}$, where $\hat{Q}$ is the $\lceil 0.9(n+1) \rceil / n$ empirical quantile of the n scores, for a nominal 90% level (Angelopoulos & Bates, 2023). Queries of one capture unit are correlated, so we report the coverage measured on the held-out evaluation split (Table 11): 88.9% to 91.1% over all listed providers. The table also gives the rate at which the reference action defers (re-measure or re-look) on the core.

Table 11: **Reference calibration by provider.** Coverage: fraction of evaluation queries whose error lies within the radius (target 90%). Mean radius: meters for depth clearance, centimeters and degrees for pose, IoU units for grounding localization. Defer: rate of re-measure or re-look among the provider's reference actions on the core.

Provider	Capability	Mean radius	Coverage (%)	Defer (%)
UniDepth	depth	0.046 m	90.4	15.0
ZoeDepth-NK	depth	0.121 m	89.3	40.9
Depth Anything V2 Large	depth	0.053 m	90.8	21.2
Metric3D v2 Large	depth	0.070 m	89.6	24.1
Metric3D v2 Small	depth	0.157 m	88.9	41.8
UniDepthV2	depth	0.038 m	91.1	9.8
MoGe-2	depth	0.041 m	90.3	10.6
VGGT	pose	5.2 cm / 1.9°	90.2	13.0
DUSt3R	pose	9.7 cm / 3.6°	89.5	39.9
MASt3R	pose	6.8 cm / 2.5°	90.6	25.3
π^3	pose	4.6 cm / 1.7°	89.8	10.9
Grounding DINO Swin-B	grounding	0.083	90.1	13.1
Grounding DINO Swin-T	grounding	0.109	89.7	26.8
OWLv2	grounding	0.141	90.9	33.1
YOLO-World-L	grounding	0.176	89.2	41.2

Coverage stays within 1.1 points of the target for every provider, including the outdoor ones (89.0–90.9%, not listed). The ground-truth decision is the action the rule takes at the true value. On the conformal half of the calibration units, 95.8% of the reference actions that commit match it (depth 96.1%, pose 95.9%, grounding 93.7%). Deferrals (re-measure or re-look) have no ground-truth counterpart and are left out of this rate; the Defer column gives how often each reference defers. Grounding is lowest because a box can overlap the target well and still belong to a neighboring instance of the same category.

D.5 Pair Construction and Statistics

Queries. A passage query is a gap between two structures that a robot could drive through, such as a door frame or the space between two pieces of furniture. We find candidate gaps from the instance labels, measure the ground-truth clearance on the mesh in a height band from 0.3 to 1.5 m, and keep

gaps visible in the query frame on both sides. The threshold is placed at $\theta = c^*(1 \pm \delta)$, where c^* is the true clearance, δ is the task limit, and the sign is drawn at random; a smaller δ puts more decisions near the threshold. A cache-reuse query pairs a query frame with up to three earlier frames of the same scan and asks whether any of them can be reused. An object-selection query names an object category in the scene, or one that is absent from it.

Filters. We drop query regions where fewer than 30% of the pixels have ground-truth depth, frames closer than 0.3 m or 15° to an already selected frame of the same scan, and pairs in which either provider failed to return an output (0.4% of candidate pairs, almost all from DUSt3R on low-overlap views).

Statistics. Table 12 gives the switch pairs of the core by treatment and task limit. Within each capability, noisier providers create more switch pairs, since a wider interval crosses a threshold more often. The two capabilities are not directly comparable: cache-reuse tolerances are tight relative to typical camera motion, so pose treatments switch more often at a given radius ratio. A looser task limit leaves fewer switch pairs. Cache reuse has no clearance threshold; its tolerance scales 0.5, 1, and 2 are counted with $\delta = 5\%$, 15%, and 30%, from tightest to loosest, wherever results are split by task limit.

Table 12: **Switch pairs on the controlled core.** Each treatment is paired with its reference on 510 units. Left: by treatment; right: by task limit (each limit covers one third of the queries). Seen: used as a training treatment by the indoor policy.

Treatment	Seen	Pairs	Switch	Rate (%)
ZoeDepth-NK	✓	510	181	35.5
Depth Anything V2 Large	✗	510	41	8.0
Metric3D v2 Large	✗	510	83	16.3
Metric3D v2 Small	✗	510	219	42.9
DUSt3R	✓	510	228	44.7
MASt3R	✗	510	64	12.5
Seen treatments		1,020	409	40.1
Unseen treatments		2,040	407	20.0
All		3,060	816	26.7

Task limit	Switch	Rate (%)
$\delta = 0.05$	341	33.4
$\delta = 0.15$	279	27.4
$\delta = 0.30$	196	19.2
All	816	26.7

On the full evaluations, 27.0% of the indoor pairs (13,342), 24.9% of the outdoor pairs (694), and 23.1% of the grounding pairs (2,516) are switch pairs. The more-reliable track pairs the reference with UniDepthV2, MoGe-2, and π^3 on the 510 core units and has 157 switch pairs (47, 39, and 71).

D.6 Human Audit of Queries and Ground Truth

Two graduate students audited a sample of queries and ground truth before we froze the benchmark. They independently checked 400 pairs drawn at random from the indoor and grounding cores (200 switch, 200 retain) and 200 further decisions whose reference margin lies within 0.03 of zero. For each item they looked at the RGB frame, the query region, and a rendering of the ground-truth mesh, and answered two questions: is the query well posed, and does the ground-truth decision match what they see? They did not see provider names or reference actions.

Their agreement was high (Cohen's $\kappa = 0.83$). After adjudication, 9 of the 400 pairs (2.3%) had a problem: five gaps partly occluded in the query view, so the clearance was measured behind an occluder; three ScanNet frames with holes in the reconstructed mesh at the gap; and one grounding target on the wrong instance. The ground-truth decision matched their judgment on 195 of the 200 near-threshold decisions (97.5%). We keep the flagged pairs; removing them changes no number in Tables 2 to 4 by more than 0.2 points. The occlusion cases led to the visibility filter for the passage queries of the full evaluation.

D.7 Replay Episodes

The replay track checks that a better single decision also helps over a short sequence. It uses 120 ScanNet evaluation scenes with recorded trajectories and the four depth treatments, which gives

480 episodes; 131 of them are switch episodes, in which the reference action differs between the reference provider and the treatment at the first step. At each step the provider runs again on the next keyframe that is at least 0.3 m away, and the agent decides again. `RE-MEASURE` at the third step resolves to `STOP`, since a robot cannot wait forever. The budget-matched run caps re-measure requests at the single-output policy's rate; results are in Appendix G.

E Packets, Task Rules, and Tool Interfaces

E.1 Packet Fields and Cue Definitions

A packet holds one measurement and a few quality cues computed around the query. The cues are cheap, defined the same way for every provider, and do not name the model. Table 13 lists them. Two kinds of cue appear. Provider cues, such as the depth spread, are computed from the provider's output and change when the provider changes. Image cues, such as edge energy, are computed from the RGB image alone. For depth and pose they use regions fixed by the query, so they are identical for every provider. They describe the scene, for example whether the query region contains a real depth edge, so they help the agent tell a rise in spread that comes with the scene from one that comes with the model. In Proposition 1 they play the role of the side cue u. Grounding's `edge` is also computed from the image alone, but on the box the provider returns, so it can change when the box changes; the two-cue analysis of Proposition 1 therefore covers depth and pose.

Table 13: **Packet fields.** Source: P = computed from the provider output; I = computed from the image only (identical across providers); I* = computed from the image only, on the region of the provider's box, so it follows the box.

Capability	Field	Definition	Source
Depth	`depth_m`	clearance or point depth from the provider's depth map (m)	P
	`spread_m`	$\mathrm{IQR}\{D_P(u) : u \in R\}/1.349$ over valid pixels of region R (m)	P
	`valid_frac`	fraction of pixels in R with finite depth inside the provider's range	P
	`edge`	mean Sobel gradient magnitude of the grayscale crop, intensities in $[0, 1]$	I
	`range_m`	median provider depth over the whole view (read by cache reuse)	P
Pose	`t_m`, `rot_deg`	translation magnitude and rotation angle to each cached view	P
	`inlier_frac`	fraction of provider correspondences kept by a RANSAC PnP refit (4 px)	P
	`reproj_px`	median reprojection residual of the kept correspondences (px)	P
	`overlap`	share of an 8×8 grid of the query view with a SIFT ratio-test match	I
Grounding	`score`	provider score of the top box (raw, not comparable across models)	P
	`score_gap`	top score minus second score	P
	`box_frac`	box area divided by image area	P
	`n_cand`	number of candidate boxes with score above 0.2	P
	`edge`	mean Sobel gradient magnitude of the box crop	I*

For a clearance query, R is the union of two vertical strips, one on each side of the gap, and `depth_m` is the distance between the two edge points after back-projection with the median depth of each strip. For a point query in the agent loop, R is a 15×15 window at the provider's working resolution. We use the interquartile range instead of the standard deviation because a few flying pixels at a depth edge would otherwise dominate the spread. Native confidence maps are left out of the default packet: only some providers expose them, and their scales differ between models.

E.2 Task Rules

Every task maps a measurement p, a calibrated radius r, a threshold θ, and a positive scale s_τ to a decision state ϕ. The margin head predicts ϕ from the packet, and a fixed rule q reads the action from the signs of ϕ. For each two-sided range the head predicts a center $\hat{m}$ and a half-width $\hat{w} = \mathrm{softplus}(\cdot)$, and the margins are $\hat{m} \mp \hat{w}$, so they are ordered by construction and $\hat{w}s_\tau$ is the predicted radius. For cache reuse both margins are then capped at 1, since in margin units neither reference margin can exceed 1 (the rule uses $\max(0, p - r)$, and $p + r \geq 0$). The cap is monotone, so the order is kept; while the upper margin stays below 1, $\hat{w}s_\tau$ is still the predicted radius. The grounding localization margin, which has one side only, is predicted directly. For passage the reference center is the reading itself, $(p - \theta)/s_\tau$. For cache reuse it depends on the

tolerance, which the reference sets from true scene depth, and for grounding on a per-provider score calibration; neither is in the packet, so the head predicts the center for every task. Table 18 reports the center and radius errors separately.

Passage (depth). With clearance estimate p and required clearance θ, $\phi = \big[(p - r - \theta)/s_\tau,\ (p + r - \theta)/s_\tau\big]$ with $s_\tau = \theta$. Both margins positive gives `PROCEED`, both negative gives `STOP`, and mixed signs give `RE-MEASURE`. Dividing by θ makes a 5 cm error count more for a 0.8 m doorway than for a 2 m corridor, which matches how much it matters. The outdoor distance check uses the same form, with p the distance to the nearest obstacle in the region and θ the required free distance.

Cache reuse (pose). For cache entry i and component $c \in \{t, R\}$, let $\overline{p}_{c,i} = p_{c,i} + r_{c,i}$ and $\underline{p}_{c,i} = \max(0, p_{c,i} - r_{c,i})$. The margins are

$$\phi_{c,i} = \Big(\frac{\theta_c - \overline{p}_{c,i}}{\theta_c},\ \frac{\theta_c - \underline{p}_{c,i}}{\theta_c}\Big). \tag{16}$$

An entry supports reuse when both upper-bound margins are positive, and it is ruled out when a lower-bound margin is negative. The action is `USE-CACHE` if at least one entry supports reuse, `REFRESH` if every entry is ruled out, and `RE-MEASURE` otherwise. The rotation tolerance is $\theta_R = 6^\circ$ times the tolerance scale. The translation tolerance is set by parallax, $\theta_t = \bar{d}\,\tan(2^\circ \cdot \text{scale})$ with $\bar{d}$ the median depth of the query view, because the same camera shift changes the view much more near a wall than across a large room. The reference takes $\bar{d}$ from ground truth, so a depth swap leaves the reference pose margins unchanged: the tolerance the task actually has does not depend on which depth model is running. The prompt gives the agent θ_R and the rule for θ_t but not its value (Box K.1), so the agent takes $\bar{d}$ from `range_m`, the median provider depth over the view. Across scenes, `range_m` follows the true room depth and should move the tolerance. Within a depth-swap pair the scene is the same, so a change in `range_m` comes from the depth model alone and should not move the pose decision. The retain term supervises exactly this case: U contains the pose coordinates of a depth-swap pair, because their reference margins stay the same while the agent's input changes. Tolerance scales are $\{0.5, 1, 2\}$.

Object selection (grounding). A per-provider isotonic map turns the raw score into a presence probability $\hat{s}$. The presence margins are $\big[(\hat{s} - r_s - 0.5),\ (\hat{s} + r_s - 0.5)\big]$ and the localization margin is $\hat{\imath} - r_\imath - 0.5$, where $\hat{\imath}$ is the predicted IoU of the top box with the target and $r_\imath$ its radius. The action is `ACT` when the lower presence margin and the localization margin are both positive, `ABSENT` when the upper presence margin is negative, and `RE-LOOK` otherwise. Grounding uses $s_\tau = 1$.

Soft rule. L_{cross} needs a differentiable version $\tilde{q}$ of q. We replace every sign test $\phi_j > 0$ by $\sigma(\phi_j / T)$ with $T = 0.1$, combine the tests with products for "and" and with $1 - \prod(1 - \cdot)$ for "or", and renormalize over the actions of the task.

E.3 Tool Interfaces inside SpatialCLI

SpatialCLI exposes four tools through JSON schemas and XML-wrapped calls (Zhou et al., 2026). We keep its interfaces, coordinate convention (integers in $[0, 999]$), and budget notices, and make two additions. Every depth, pose, and locate result carries the quality fields of Table 13, and the call may carry an optional `check` that states what the reading should decide. Box E.1 shows the depth registration; the pose and locate registrations change in the same way, and Box E.2 shows their returns. Segment is unchanged.

Box E.1: Depth tool registration with quality fields and checks

```
{
  "name": "query_depth",
  "description": "Metric depth (camera-axis Z, meters) at one or more 2D points. Use it
     for which-is-closer, distance, and clearance questions. Each result also reports
     quality fields for a 15x15 window around the point: spread_m (robust spread of
     depth), valid_frac (share of valid depth pixels), and edge (image edge energy). The
     model behind this tool can change between deployments. Add a check when the reading
     is meant to decide a relation.",
  "parameters": {
```

```
    "type": "object",
    "properties": {
      "points": {
        "type": "array", "minItems": 1, "maxItems": 8,
        "items": {"type": "array", "items": {"type": "integer"},
                  "minItems": 2, "maxItems": 2},
        "description": "Points [x, y] in the shared 0-999 image coordinates."
      },
      "image_indices": {
        "type": "array", "items": {"type": "integer", "minimum": 1},
        "description": "Optional 1-based image numbers."
      },
      "check": {
        "type": "object",
        "description": "Optional. {\"type\": \"depth_order\", \"a\": \"point_1\", \"b\":
      \"point_2\"} tests whether point a is closer than point b; {\"type\":
      \"depth_at_least\", \"a\": \"point_1\", \"value_m\": 1.5} tests a distance."
      }
    },
    "required": ["points"]
  }
}

Example:
<tool_call>
{"name": "query_depth", "arguments": {"points": [[318, 604], [702, 551]],
     "image_indices": [1], "check": {"type": "depth_order", "a": "point_1", "b":
     "point_2"}}}
</tool_call>

Possible return:
{"count": 2, "result": [{"point_2d": [318, 604], "depth": 1.19, "spread_m": 0.14,
     "valid_frac": 0.81, "edge": 0.29}, {"point_2d": [702, 551], "depth": 1.30,
     "spread_m": 0.09, "valid_frac": 0.93, "edge": 0.18}]}
```

Box E.2: Pose and locate returns with quality fields

```
Pose (camera motion from image 1 to image 2), check {"type": "camera_motion", "axis":
     "x", "sign": "left"}:
{"motion": {"yaw_deg": 12.4, "pitch_deg": -1.1, "roll_deg": 0.2, "t_m": [-0.21, 0.02,
     0.35],
  "inlier_frac": 0.63, "reproj_px": 2.8, "overlap": 0.54}}

Locate, check {"type": "present", "query": "red mug"}:
{"count": 1, "result": [{"bbox_2d": [532, 470, 628, 675], "point_2d": [580, 572],
  "score": 0.47, "score_gap": 0.21, "box_frac": 0.020, "n_cand": 3, "edge": 0.24}]}
```

E.4 Evidence Checks and Scales

A check tells the loop which quantity decides the question, so that one margin head can serve open-ended questions. Each check sets the measurement p, the threshold θ, and the scale s_τ as follows.

- `depth_order` (a closer than b): $p = d_b - d_a$, $\theta = 0$, $s_\tau = \max\big((|d_a| + |d_b|)/2,\ 0.1\,\mathrm{m}\big)$. Depth errors grow with distance, so we scale by the mean depth; the 0.1 m floor keeps the margin finite for points very close to the camera. `depth_at_least` uses $p = d_a$, $\theta =$ `value_m`, $s_\tau = \theta$.
- `camera_motion` and `camera_position`: p is the signed translation (or yaw) along the stated axis, $\theta = 0$, and $s_\tau = 0.1$ m for translation and 10° for rotation.
- `present`: the presence margins of Appendix E.2; localization uses the IoU margin. Both use $s_\tau = 1$.

The reference radius for a depth-order check is calibrated directly on the residual of the difference $d_b - d_a$, not on each depth separately. Two readings from one depth map share much of their error (for example a global scale error), so the error of their difference is smaller than the two separate errors would suggest.

After each checked return the policy emits its anchor token, the margin head predicts the two margins, and the rule turns them into a verdict that the loop appends to the observation: `USE` when both margins have the same sign (the check holds or fails clearly), `RE-QUERY` when the interval

crosses zero and budget remains, and UNUSABLE when it crosses zero and no budget is left. After UNUSABLE the agent answers from its remaining evidence.

A worked check. For "Is A closer than B?", a first depth call returns $d_A = 1.19\,\text{m}$ and $d_B = 1.30\,\text{m}$, so $p = 0.11\,\text{m}$ and $s_\tau = 1.245\,\text{m}$. The spread at A is large (0.14 m) and the margin head predicts margins $(-0.040, +0.217)$, which correspond to a radius of 0.16 m. The interval $[-0.05, 0.27]$ m contains zero, so the verdict is RE−QUERY, and the agent queries a point lower on object A, away from its boundary. The second call returns $d_A = 1.21\,\text{m}$ and $d_B = 1.33\,\text{m}$ with smaller spreads; now $p = 0.12\,\text{m}$, $s_\tau = 1.27\,\text{m}$, the predicted margins are $(+0.039, +0.150)$, and the verdict is USE: A is closer.

F Baseline Implementations

Matched controls. Every policy that predicts margins starts from Qwen3-VL-8B-Instruct and uses the same LoRA configuration (rank 16, scaling 32, dropout 0.05 on the query and value projections), the same margin head, the same task rule, and the same sequence limit (1,024 tokens on CapSwap-Bench, 4,096 for the adapters used inside SpatialCLI). The single-output control runs the warm-up objective L_{warm} for 214 steps and then for another 398 steps on the packets that make up CapAgent's pairs. It reads the same packets in the same order, without the pair losses. All policies with a margin head use the checkpoint rule of Appendix C.1. *Different-scene pairs* (Table 3) keep CapAgent's loss, schedule, and batch balance, but B's swapped query comes from another capture unit, drawn at random among training queries of the same task and threshold, and is run through the other training provider. B keeps A's second query and its reference packet, so U and the retain term are as in CapAgent; switch and retain labels come from the two reference actions.

Action-trained baselines. *Trajectory SFT* supervises the full response, including the action token, with no margin head, for the same 612 steps. *SFT + GRPO* continues from the trajectory SFT checkpoint for 300 GRPO updates (Shao et al., 2024) with group size 8, learning rate 1×10^{-6}, clip bounds 0.20 and 0.28, and no KL term. The reward is 1 when the action matches the reference action and 0 otherwise, minus 0.5 for an unsupported commitment. Without this penalty, GRPO raised validation accuracy by committing more often.

Paired baselines. *Paired-input SFT* places both packets of a pair in one prompt and supervises both actions. At test time the second slot repeats the single packet; leaving the slot empty was 1.7 switch points worse on validation, because the model had never seen an empty slot during training. *DPO* (Rafailov et al., 2023) starts from trajectory SFT and runs 398 steps with $\beta = 0.1$. For each switch pair, the correct action for B's output is preferred and A's action, the stale one, is dispreferred. *DPO + margin head* adds L_{state} on both outputs; the preference gradient does not flow into the head, so the head stays a margin regressor.

GAR. GAR (Zhu et al., 2025) fits differences of predictions to differences of labels over all pairs of a batch. It has three terms: a pointwise MAE; a squared difference term, which by their Theorem 1 equals the variance of the pointwise errors and is therefore computed in linear time as an MSE on batch-centered predictions and labels; and a normalized-difference term ($p = 2$), which equals one minus the Pearson correlation of predictions and labels. The three are combined with GAR's log-power-mean, whose exponent we choose from $\{0.1, 1, 10\}$ on validation (1). We apply GAR to our margins, one margin coordinate at a time, over the packets of the batch that have that coordinate, so packets of different tasks are never compared on a coordinate that one of them lacks. L_{warm} stays on every packet, and the GAR objective is added with weight 0.5, chosen from $\{0.1, 0.5, 1.0\}$ on validation. GAR follows the schedule of the single-output control, so in its last 398 steps each batch holds the 32 outputs of 16 same-scene pairs; 16 of the 496 pairs in a batch share a scene, and the others change it. This is the GAR row of Tables 2 and 1. Its within-pair slope is 0.32, against 0.30 for single-output training and 0.57 for CapAgent.

Three further runs are ablations of CapAgent. *GAR loss, same-scene pairs* replaces L_{shift}, L_{cross}, and L_{retain} by GAR's difference and normalized-difference terms, combined with the same log-power-mean and computed with explicit pair sums over the 16 same-scene pairs of each batch only. It drops GAR's pointwise MAE, since L_{warm} already fits each output. The terms run over every mar-

gin coordinate the two packets share, including the untouched coordinates U, whose target change is zero; there the squared difference term acts as a squared retain term. For a coordinate whose target changes have zero variance over the 16 pairs, as on U, the normalized-difference term is skipped, and its normalization uses $\epsilon = 10^{-6}$. *GAR loss + cross + retain* adds our cross and retain terms with our weights ($\lambda_x = 0.5$, $\lambda_r = 0.25$), so U is held both by GAR's squared difference and by our retain term. *GAR loss on J + cross + retain* runs GAR's two terms only over the coordinates in J, as our shift loss does, and U is held by our retain term alone. It reaches 82.4% switch accuracy, 2.3 points $[0.7,\ 3.9]$ below CAPAGENT.

Domain generalization baselines. Both methods treat the training providers of a capability as environments and use the same data and schedule as the single-output control. DANN removes provider information from the anchor state; V-REx leaves the representation free and asks the providers' risks to be equal. *DANN* (Ganin et al., 2016) attaches a provider classifier to each anchor state (a two-layer MLP with 1,024 hidden units, two classes per capability) through a gradient-reversal layer. The reversal coefficient follows the original ramp $2/(1 + e^{-10p}) - 1$ over training progress p, scaled by a maximum value chosen from $\{0.1, 0.3, 1.0\}$ on validation. We use 0.3; at 1.0 the margin head became unstable and validation accuracy fell by 4.1 points. *V-REx* (Krueger et al., 2021) adds the variance of the per-provider L_{state} risks to the loss. The penalty weight is chosen from $\{1, 10, 100\}$ (we use 10) and is switched on after 100 steps, following the annealing used in the original work.

Rules and recalibration. The *fixed-radius rule* uses one conformal radius per task, fitted on the reference provider's calibration errors without the regressor, and applies the task rule to the packet's measurement. It ignores the quality cues, so it can only change its decision when the measurement itself moves. On a switch pair it therefore tends to be right on the reference output and wrong on the replacement, which puts its Switch accuracy near one half. *Recalibration* fits a new radius for each unseen provider on N labelled capture units: a quantile regressor on the packet cues, followed by a split-conformal correction on half of the N units, and then the reference rule. It uses ground-truth labels of the new provider.

Baselines inside SpatialCLI. All trained 8B agents in Table 1 (Tool-use SFT-8B, Single-output-8B, GAR-8B, and CAPAGENT-8B) train on the same data (Appendix C.1), receive the same prompt additions and quality fields, and have the same limit of ten tool calls. *Tool-use SFT* supervises the response text, including the stated check and its verdict, without a margin head. GAR-8B uses GAR's published objective. The SpatialCLI rows use SpatialCLI's original prompt and tool returns, so Single-output-8B, not SpatialCLI-8B, is the matched control for paired training. The loss ablation GAR loss + cross + retain, which keeps CAPAGENT's pairs and terms and uses GAR's pair loss, reaches 69.8 in the loop (Section 5.4). *Frontier models* (GPT-5.6 Sol and Qwen3.5-397B-A17B) are called through their APIs with SpatialCLI's agent prompt and the same four tools. In the swapped-tool runs their tool returns carry the same quality fields as ours; their original-tool scores are SpatialCLI's reported runs, without these fields. The call-capped CAPAGENT-8B and the retrained SpatialCLI-8B are described with Table 15.

G ADDITIONAL EXPERIMENTS

The subsections follow the order of Section 5.

G.1 STANDARD BENCHMARKS INSIDE SPATIALCLI

Original-tool scores. Table 1 reports each agent after the swap together with its change from the original tools. Table 14 gives the original-tool scores behind those changes. VGGT and Grounding DINO are the pose and grounding references of our training pool; Depth Anything 3 is not in it. With the original tools, all trained agents stay within 0.3 points of SpatialCLI-8B on average, and CAPAGENT-8B is 1.4 points lower on camera motion (Case 4, Appendix L).

Other tool swaps. The main swap in Table 1 replaces three models at once. Table 15 replaces one tool at a time and adds a swap to newer models (Depth Pro (Bochkovskii et al., 2025), π^3,

Table 14: **Standard benchmarks with the original tools** (%), for every agent of Table 1. Frontier-model and SpatialCLI rows are the values reported by Zhou et al. (2026). Bold: best among 8B agents.

	SpatialCLI		MMSI			BOPASK		
Agent	Bench	MindCube	Motion-Cam	Pos-Cam-Cam	DA-2K	Traj.	ObjRrr	Avg.
GPT-5.6 Sol	72.9	72.1	54.1	62.4	86.1	52.8	49.5	68.1
Qwen3.5-397B-A17B	79.8	67.6	46.0	54.8	91.9	57.7	44.7	68.2
SpatialCLI-8B	91.3	84.6	39.2	53.8	91.8	55.2	48.5	73.2
Tool-use SFT-8B	91.0	84.5	**39.4**	53.2	91.6	55.1	48.4	73.0
Single-output-8B	90.9	84.4	39.2	52.7	91.5	55.0	48.2	72.9
GAR-8B	91.1	84.6	39.0	53.9	91.7	55.2	48.4	73.1
CAPAGENT-8B	**91.5**	**84.9**	37.8	**54.8**	**92.0**	**55.4**	**48.9**	**73.4**
SpatialCLI-27B	91.7	85.5	52.7	53.8	91.9	58.2	55.6	75.9

and OWLv2). The depth swap costs the most (3.8 points), which fits the benchmarks: SpatialCLI-Bench and DA-2K both depend on depth comparisons. With the newer models SpatialCLI-8B loses 1.4 points. They move reliability in both directions: π^3 has 0.88 times the radius of VGGT, and OWLv2 1.70 times that of Grounding DINO (Table 9). CAPAGENT-8B recovers most of this loss and calls tools slightly less often than SpatialCLI-8B (2.4 against 2.5), because it accepts more first measurements from Depth Pro and π^3.

Call cap and retraining. CAPAGENT-8B makes 3.0 calls per question after the joint swap, against 2.6 for SpatialCLI-8B. Capped on every question at the number of calls SpatialCLI-8B made on it, it reaches 69.7, above every other 8B agent in Table 1. Retraining is the direct alternative: we regenerate SpatialCLI's SFT trajectories and RL rollouts with the three replacement tools behind the same interfaces, keeping its training questions, and rerun its SFT and RL recipe from Qwen3-VL-8B-Instruct with the same steps and hyperparameters. The retrained SpatialCLI-8B reaches 71.9 with 2.5 calls per question, 1.3 points above CAPAGENT-8B.

Table 15: **Other tool swaps.** Avg: macro average over the five benchmarks of Table 1 (SpatialCLI-8B with original tools: 73.2). Calls: tool calls per question. Replacements: DA3 → Metric3D v2 Small, VGGT → MASt3R, Grounding DINO → YOLO-World-L; newer: DA3 → Depth Pro, VGGT → π^3, Grounding DINO → OWLv2.

	Avg↑				Calls	
Swapped tools	SpatialCLI-8B	Single-output-8B	GAR-8B	CAPAGENT-8B	SpatialCLI-8B	CAPAGENT-8B
Depth only	69.4	70.2	70.8	**71.8**	2.5	2.8
Pose only	70.9	71.5	71.7	**72.2**	2.5	2.7
Grounding only	71.6	71.9	72.1	**72.5**	2.5	2.6
All three (Table 1)	66.5	67.9	68.8	**70.6**	2.6	3.0
Newer models	71.8	72.0	72.2	**72.9**	2.5	2.4
All three, other budgets and training						
CAPAGENT-8B, SpatialCLI-8B's calls per question	–	–	–	69.7	–	2.6
SpatialCLI-8B retrained with the new tools	71.9	–	–	–	2.5	–

G.2 FULL CONTROLLED RESULTS

Table 16 extends Table 2 with plain DPO, the ablation GAR loss + cross + retain of Section 5.4 and its variant with GAR's terms on J only, 95% scene-bootstrap intervals, and *New*, the accuracy on the replacement's own output in unseen switch pairs. The reference output of such a pair was seen in training and the replacement output was not, so New isolates the harder half and separates the methods more than the pair average. CAPAGENT reaches 66.1% against 39.3% for single-output training and 61.2% for GAR loss + cross + retain, so most of CAPAGENT's gain comes from reading the unseen model's output correctly. The domain generalization baselines move New by only 3.5–4.7 points.

G.3 GROUND-TRUTH OUTCOMES

Table 17 scores actions against scene annotations instead of reference actions. CAPAGENT defers slightly more often than the single-output policy, so the budget row caps each task's total deferrals

Table 16: **Controlled core, all baselines.** Reference-action agreement (%) on 3,060 pairs (816 switch, 407 of them unseen). Brackets: 95% scene-bootstrap intervals. New: accuracy on the replacement's output in unseen switch pairs. †Reference rule that ignores the cues; it keeps most supported actions by never reacting to a swap and is not ranked.

Method	Acc↑	Unsup.↓	Switch↑	Unseen↑	New↑	Retain↑
Fixed-radius rule†	82.2	6.4	47.4 [45.9, 48.9]	46.7 [44.6, 48.8]	9.6	94.8
Trajectory SFT	78.1	5.6	54.1 [51.3, 56.9]	41.3 [37.6, 45.0]	27.5	86.8
SFT + GRPO	79.3	5.1	57.6 [54.9, 60.3]	44.0 [40.3, 47.7]	30.7	87.2
Single-output margin	83.5	4.1	68.4 [65.9, 70.8]	52.8 [49.1, 56.4]	39.3	89.0
DANN	83.7	4.0	70.3 [67.8, 72.7]	55.2 [51.6, 58.8]	42.8	88.6
V-REx	84.2	3.9	71.2 [68.7, 73.6]	56.0 [52.4, 59.6]	44.0	88.9
Paired-input SFT	80.4	5.0	63.9 [61.1, 66.6]	48.6 [44.9, 52.3]	35.1	86.4
DPO	78.2	6.1	66.2 [63.5, 68.8]	50.1 [46.4, 53.8]	41.8	82.5
DPO + margin head	83.4	4.9	74.6 [72.1, 77.0]	58.4 [54.8, 62.0]	47.4	86.6
GAR	85.5	3.7	73.9 [71.3, 76.4]	59.1 [55.5, 62.6]	49.4	89.7
Pairs and terms of CAPAGENT*, pair loss of GAR*						
GAR loss + cross + retain	89.3	2.5	81.8 [79.3, 84.2]	69.2 [65.7, 72.6]	61.2	92.0
GAR loss on J + cross + retain	89.3	2.4	82.4 [79.9, 84.8]	69.9 [66.4, 73.3]	62.2	91.8
CAPAGENT	**90.3**	**2.2**	**84.7** [81.9, 87.4]	**72.9** [69.5, 76.2]	**66.1**	**92.3**

(re-measure or re-look) at the single-output count. Once a task's budget is used up, depth and pose choose the action from the predicted interval center, and grounding uses the predicted presence sign and the highest-scoring candidate. Under this cap CAPAGENT keeps 71–83% of its reduction in false proceeds, invalid cache reuse, and acting on a wrong or absent target, the errors that commit on weak evidence, and 57–71% of its reduction in unnecessary stops, refreshes, and missed targets. Most of the gain therefore comes from better decisions. The last row applies the reference actions themselves, which err on 0.4–1.1% of decisions.

Table 17: **Ground-truth outcomes on the controlled cores** (% of decisions; indoor core for depth and pose, grounding core for grounding). FP false proceed, FS unnecessary stop, IR invalid cache reuse, VR unnecessary refresh, WA act on a wrong or absent target, MP missed target; RM/RL re-measure or re-look rate.

	Depth			Pose			Grounding		
Policy	FP↓	FS↓	RM	IR↓	VR↓	RM	WA↓	MP↓	RL
Fixed-radius rule	4.9	3.1	14.0	5.6	3.4	15.2	9.4	4.0	12.4
Trajectory SFT	4.4	2.6	19.8	4.7	2.9	19.1	8.1	3.6	18.0
Single-output	3.0	1.9	21.7	3.5	2.3	21.0	6.2	2.8	20.6
CAPAGENT (budget)	2.0	1.4	21.7	2.2	1.7	21.0	3.8	2.4	20.6
CAPAGENT	**1.6**	**1.2**	23.1	**1.9**	**1.4**	22.6	**3.3**	**2.1**	22.9
Reference actions	*0.8*	*0.4*	*23.5*	*0.9*	*0.6*	*22.8*	*1.1*	*0.7*	*23.4*

G.4 THE AGENT'S OWN RANGES

Table 18 compares the agent's own ranges with the reference and with ground truth.

Center and radius. For passage the reference center is the reading, so an error in the predicted center is an error in reading the measurement, and an error in the radius is an error in judging reliability (panel a). Pairing leaves the center error almost unchanged and cuts the radius error on unseen outputs from 3.9 to 2.3 cm.

Coverage against ground truth. Panel b counts how often the agent's own range contains the true value. The single-output agent's ranges are too narrow on less reliable replacements (72.4% coverage, radius 0.71 times the reference's) and too wide on more reliable ones (96.3%, 1.38 times). Pairing mostly repairs the first case (85.2%, 0.87) and only part of the second (1.24), which matches the weaker transfer toward more reliable models (Table 21). Switch outputs sit near a threshold by construction, and there even the reference covers only 85.9%.

Table 18: **The agent's own ranges** on the controlled core. (a) Passage outputs: mean absolute error of the predicted center and radius against the reference (cm), and switch accuracy on the 524 depth switch pairs (%). (b) Depth and pose outputs: share of outputs whose true value lies in the range (%; nominal 90), and mean predicted radius relative to the reference's. More reliable: the three more reliable replacements of Table 21 on the core units. Unseen switch: replacement outputs in unseen switch pairs.

(a) Center and radius of passage ranges

	Center error↓		Radius error↓		
Policy	Seen	Unseen	Seen	Unseen	Depth switch↑
Single-output margin	0.8	0.9	1.5	3.9	64.0
GAR loss + cross + retain	0.8	0.9	1.4	2.7	78.0
CAPAGENT	**0.7**	**0.8**	**1.3**	**2.3**	**81.5**

(b) Coverage of the true value and width of the range

	Coverage (%)				Radius / reference	
Policy	Seen	Unseen	More reliable	Unseen switch	Unseen	More reliable
Reference	*89.9*	*90.0*	*90.4*	*85.9*	*1.00*	*1.00*
Single-output margin	88.6	72.4	96.3	58.7	0.71	1.38
GAR loss + cross + retain	89.0	82.1	95.1	74.8	0.82	1.27
CAPAGENT	89.3	85.2	94.7	79.6	0.87	1.24

G.5 EXTRA INFORMATION AT DEPLOYMENT

Table 19 gives two kinds of extra information that CAPAGENT does not use. Adding the provider's name to the packet raises the single-output policy's switch accuracy on familiar providers but lowers it slightly on unseen ones (52.8% to 51.8%): the name is useful only for models already seen in training. Recalibration fits a new radius for each unseen provider from N labelled capture units. It passes CAPAGENT between 20 and 50 labelled units per provider and reaches 88.5% at 100.

Table 19: **Switch accuracy with extra information at deployment** (%). Recalibration uses N labelled capture units of each unseen provider and reads only the packet cues.

System	Extra information	All↑	Unseen↑
Single-output margin	none	68.4	52.8
Single-output margin + ID	provider name	71.3	51.8
CAPAGENT	none	84.7	72.9
CAPAGENT + ID	provider name	85.1	72.4
Recalibration, $N = 5$	5 labelled units	–	60.2
Recalibration, $N = 20$	20 labelled units	–	71.5
Recalibration, $N = 50$	50 labelled units	–	81.0
Recalibration, $N = 100$	100 labelled units	–	88.5

G.6 PARTS OF THE PAIR LOSS

Table 20 removes the parts of the pair loss one at a time, always on same-scene pairs. Shift + cross switches most often, 87.9%, but changes too many decisions: retention falls to 71.0% and untouched pose decisions to 44.2%. Without the cross term, shift stays below GAR's loss on the same pairs (76.5% alone and 73.8% with retain, against 78.8%). The retain term costs 3.2 switch points and restores retention to 92.3%. Replacing shift by a loss on the direction of the change alone, or by a relational-distillation loss on the distance between the two margin vectors (Park et al., 2019), lowers every column; fitting the signed size of the change is worth 8.5 and 10.5 switch points over these two. Reversing the paired targets drops switch accuracy to 29.4% [26.1, 32.7] (24.0% unseen), far below single-output training.

Table 20: **Parts of the pair loss** on the controlled core (%), same-scene pairs. Direction and distillation replace shift in the full loss; reversed trains the full loss on reversed paired targets. Untouched: accuracy on the capability whose provider did not change.

Pair loss	Switch↑	Retain↑	Untouched↑	Unsup.↓
Shift	76.5	70.4	43.0	6.1
Shift + cross	**87.9**	71.0	44.2	5.8
Shift + retain	73.8	91.2	80.1	2.7
Cross + retain	72.5	91.5	81.0	2.9
Shift + cross + retain (CAPAGENT)	84.7	**92.3**	**82.4**	**2.2**
Direction instead of shift	76.2	90.8	79.3	3.0
Distillation instead of shift	74.2	88.1	76.9	3.6
Reversed paired targets	29.4	70.6	–	9.2

G.7 RESULTS BY REPLACEMENT PROVIDER

Table 21 splits the core by treatment provider and adds three replacements that are more reliable than the reference. Both policies are strong on the two treatments seen in training (ZoeDepth-NK and DUSt3R), and single-output training already reaches 83.9% there. Its failure is specific to new providers, where it drops to 52.8%. This matches the identity probe in Section 5.5: part of what single-output training learns is which provider it is looking at. The hardest unseen provider is the one closest to the reference. Depth Anything V2 Large has a radius only 1.15 times the reference, so its switch pairs sit just past a threshold and the required change in margins is small. CAPAGENT's accuracy on it is 62.2%, against 78.8% on Metric3D v2 Small, whose large change is easier to detect. Within each capability, the number of switch pairs grows with the radius ratio, since a wider interval crosses a threshold more often.

Upgrades are a common reason to swap a model, so the bottom block pairs the reference with UniDepthV2, MoGe-2, and π^3, whose calibrated radius is below the reference. Here the correct change is usually toward committing: the reference output crosses a threshold and the better model's does not. CAPAGENT improves switch accuracy from 43.3% to 59.9% and cuts needless re-measuring on the new output from 28.4% to 16.9%. Accuracy stays well below the core, and New is only 45.9%: the agent's ranges on these models stay 1.24 times as wide as the reference's (Table 18b).

Table 21: **Switch accuracy by replacement provider.** Top: the six treatments of the controlled core. Bottom: replacements more reliable than the reference, paired with it on the 510 core units. Ratio: calibrated radius relative to the reference provider. Switch: pair-averaged switch accuracy (%). New: accuracy on the replacement's own output in switch pairs (%). S: single-output margin; C: CAPAGENT. Group rows are weighted by switch pairs.

				Switch↑		New↑	
Treatment	In training	Ratio	Switch pairs	S	C	S	C
ZoeDepth-NK (depth)	✓	2.62	181	84.5	95.9	80.1	94.5
DUSt3R (pose)	✓	1.87	228	83.3	97.1	78.9	96.1
Depth Anything V2 Large (depth)	✗	1.15	41	47.6	62.2	31.7	53.7
Metric3D v2 Large (depth)	✗	1.52	83	50.0	66.9	36.1	59.0
Metric3D v2 Small (depth)	✗	3.41	219	55.5	78.8	42.5	72.6
MASt3R (pose)	✗	1.31	64	50.8	67.2	37.5	60.9
Seen treatments			409	83.9	96.6	79.5	95.4
Unseen treatments			407	52.8	**72.9**	39.3	**66.1**
All			816	68.4	**84.7**	59.4	**80.8**
More reliable than the reference (not in the core)							
UniDepthV2 (depth)	✗	0.82	47	42.6	58.5	23.4	44.7
MoGe-2 (depth)	✗	0.90	39	44.9	61.5	25.6	48.7
π^3 (pose)	✗	0.88	71	43.0	59.9	23.9	45.1
More reliable			157	43.3	**59.9**	24.2	**45.9**

G.8 Full Evaluations, Other Tracks, and Replay

Table 22 applies the frozen policies to the full indoor, outdoor, and grounding evaluations. Outdoor and grounding use separately trained policies with the same recipe. The gain holds on all three. It is smallest on grounding (12.8 switch points), where the presence score of a detector is already a fairly direct reliability signal and the single-output policy starts higher. The indoor gain on the full set (17.6 points) is close to the core gain (16.3). On the 480 replay episodes (Appendix D.7), CAPAGENT raises final-step accuracy from 79.8% to 88.6% and switch-episode accuracy from 61.7% to 80.9%, and unsupported commitments fall from 5.4% to 2.3%. With re-measures capped at the single-output rate (0.57 per episode, against 0.66 uncapped), it reaches 87.1% and 78.4%.

Table 22: **Full evaluations** (%). Indoor: 49,416 pairs; outdoor: 2,788; grounding: 10,890. New: accuracy on the replacement's output in unseen switch pairs.

			Switch↑			
Split	Policy	New↑	all	unseen	Retain↑	Unsup.↓
Indoor	Single-output	36.6	60.2	48.8	88.1	5.0
	CAPAGENT	**61.8**	**77.8**	**69.7**	**91.9**	**2.9**
Outdoor	Single-output	44.8	55.9	53.8	87.6	5.2
	CAPAGENT	**61.6**	**71.8**	**69.5**	**91.4**	**3.1**
Grounding	Single-output	44.1	61.4	58.6	87.5	8.3
	CAPAGENT	**61.5**	**74.2**	**70.9**	**90.8**	**4.5**

G.9 A Replacement Whose Errors the Cues Do Not Show

CAPAGENT reads reliability from the cues, so it can follow an error only if the error changes them. To test this limit we build a replacement whose error the cues hide: UniDepthV2 with its depth map multiplied by 1.06. Every clearance reads 6% wide. Spread grows by the same 6%, which is small next to the spread differences between models, and valid fraction and edge energy do not change. We calibrate its reference like any other provider, which widens its radius from 0.038 m to 0.086 m, and pair it with UniDepth on all indoor depth queries of the evaluation split (8,236 pairs, 2,217 of them switch pairs). Most switch pairs turn a commitment on UniDepth's output into a re-measure on the scaled one.

Table 23 scores the scaled model's own outputs in these switch pairs, since the other output is UniDepth's and does not change. Every trained policy predicts a radius close to that of the unscaled model and keeps committing on the wide readings: accuracy stays within 7.1 points of the fixed-radius rule, and about one decision in ten is an unsupported commitment. Recalibration on 20 labelled units sees the bias in the residuals and reaches 64.3%.

Table 23: **A replacement whose error the cues hide**: UniDepthV2 with depth scaled by 1.06, paired with UniDepth on all indoor depth queries of the evaluation split (%). New: accuracy on the scaled model's outputs in switch pairs. Radius: mean predicted radius on those outputs (reference: 0.086 m). †Ignores the cues; not ranked. Recalibration uses 20 labelled units of the scaled model.

Policy	New↑	Unsup.↓	Radius (m)
Fixed-radius rule†	8.1	10.6	0.046
Single-output margin	11.9	9.8	0.035
GAR loss + cross + retain	13.4	9.5	0.036
CAPAGENT	15.2	9.1	0.037
Recalibration, $N = 20$	64.3	3.6	0.081

G.10 Failure Modes

We labelled a random sample of CAPAGENT's switch errors on the full indoor evaluation for one seen and two unseen depth providers (Table 24). For the replacement closest to the reference (Depth Anything V2 Large, $1.15\times$), three quarters of the errors move the margins in the right direction

but stop short of the threshold: the change is small and the model underestimates it. For the most unreliable replacement (Metric3D v2 Small, 3.41×), 43.4% of the errors answer `STOP` where the reference re-measures: the reading itself has fallen below the threshold, and the predicted range is too narrow to reach back above it. These errors come from a radius that is too small or, in 4 of the 23, from a center more than 1 cm away from the reading. The agent's mean radius on this provider is 0.124 m against a calibrated 0.157 m, close to the widest training radius (ZoeDepth-NK, 0.121 m). Together with the 32.1% of errors that stop short of the threshold, most errors on this provider predict too narrow a range. Both failure types point to the same limit: CAPAGENT learns the size of a change well within the range of changes it was trained on and less well outside it.

Table 24: **CAPAGENT's switch errors by type**, from a random sample of errors on the full indoor evaluation. Ratios: calibrated radius relative to the reference.

Failure mode	Seen, 2.62× ($n = 57$)	Near, 1.15× ($n = 33$)	Far, 3.41× ($n = 53$)
Right direction, no crossing	57.9%	75.8%	32.1%
Wrong direction	29.8%	15.2%	22.6%
Opposite commitment	8.8%	6.1%	43.4%
Other	3.5%	3.0%	1.9%

G.11 SCALING WITH DATA, RELIABILITY CHANGE, RETAIN WEIGHT, AND CALL BUDGET

Figure 5 collects four sweeps. (a) With 25 paired scenes CAPAGENT already beats single-output training on 200 scenes (75.6% against 68.4%); the curves are still rising at 200 scenes for all three methods. (b) Across ten domain–provider settings, each training its own single-output and paired policies with one treatment, the gain from pairing grows with how much less reliable the replacement is. (c) The retain weight trades switch accuracy for accuracy on untouched capabilities; without the retain term, untouched accuracy falls to 45.1%, and we pick $\lambda_r = 0.25$ by the harmonic mean. (d) With swapped tools, CAPAGENT-8B's lead over Single-output-8B appears once two or more calls are allowed. With one call, no policy can act on a crossing interval, and the three agents score within 0.8 points of each other (60.1 to 60.9).

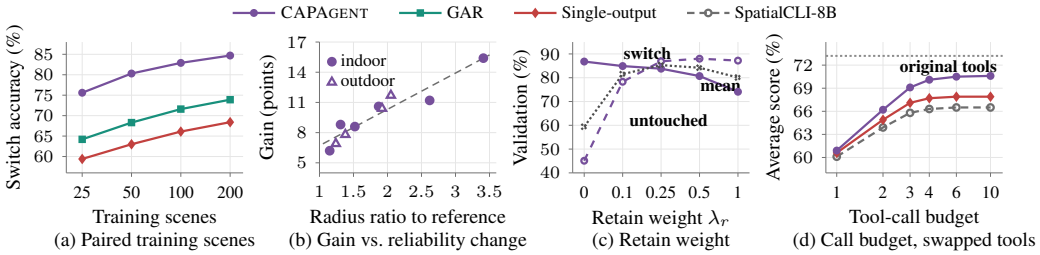


Figure 5: **Scaling analyses.** (a) Switch accuracy on the controlled core against the number of paired training scenes. (b) Pairing gain (switch points) against the calibrated radius ratio of the replacement, for ten domain–provider settings (indoor and outdoor depth, indoor pose), each with its own single-output and paired policies; the dashed line is a least-squares fit. (c) Switch accuracy, untouched-capability accuracy, and their harmonic mean on validation scenes for different retain weights. (d) Macro-average score on the five standard benchmarks with swapped tools against the tool-call budget per question; the dotted line is SpatialCLI-8B with its original tools.

G.12 POLICY CLASSES

Pairing helps every policy class we tried (Table 25), from a 0.2M-parameter tabular regressor to two 8B vision-language models, by 10.5 to 16.3 switch points. The small models are competitive on this controlled task because the packet is a short vector.

Pairing also works in a separate module. In SpatialCLI's loop we replace the margin head of Single-output-8B by the paired packet-MLP of Table 25, which reads the numbers of one return and the stated check, and keep the planner unchanged. With swapped tools it reaches 69.0, between Single-output-8B (67.9) and CAPAGENT-8B (70.6). The module does not see the image or the question,

Table 25: **Policy classes** (switch accuracy on the controlled core, %). Params: trainable parameters.

Policy	Params	Single	Pair	Gain
Tabular regressor	0.2M	68.1	79.6	+11.5
Packet-MLP	3.1M	66.3	76.8	+10.5
Qwen2.5-VL-3B	6.8M	64.8	78.7	+13.9
Qwen3-VL-8B-Instruct	12.6M	68.4	84.7	+16.3
SpatialCLI-8B checkpoint	12.6M	70.2	**86.1**	+15.9

which the 8B head reads through the anchor state; the difference is largest on SpatialCLI-Bench (83.4 against 86.2), where glass, thin frames, and partly visible objects decide how far a reading can be trusted.

G.13 Cost

Pairing changes which packets are compared in the loss, not how many are processed, so CAPAGENT trains at the same cost as the single-output policy (Table 26). GAR loss + cross + retain is slightly slower because its normalized-difference term needs statistics over all 16 pairs of a batch before the backward pass. The 16 pairs are spread over eight cards, two micro-batches per card, so every card keeps both forward passes in memory until the head outputs are gathered (Appendix C.3). At deployment CAPAGENT reads one packet, and the action is available from the margin head before any explanation is generated. Inside SpatialCLI's loop the extra cost is the additional tool calls: 0.4 more calls per question than SpatialCLI-8B, which adds 3.9 seconds to its 31.1 seconds per question.

Table 26: **Training and inference cost** on RTX 4090 GPUs. GPU-hours: training the adapters used inside SpatialCLI (867 steps on eight cards, including validation). Head-only: time to the action from the margin head per packet on one card. Response: full generated response per packet. Loop: mean wall-clock time per SpatialCLI-Bench question with swapped tools.

Policy	Trainable	GPU-hours	Head-only (s)	Response (s)	Loop (s)
Single-output margin (8B)	12.6M	43.2	0.21	6.1	31.9
GAR loss + cross + retain (8B)	12.6M	44.4	0.21	6.3	34.1
CAPAGENT (8B)	12.6M	43.2	0.21	6.3	35.0

H Tool Calls Before and After a Swap

This section shows what a swap looks like from inside the agent loop. In each pair below, the same call with the same arguments is sent on the same input, once to the original backing model (blue, left) and once to the replacement (orange, right). The return format is identical, so nothing in the interface signals the swap; only the numbers and the quality fields change. The verdict line under each return is what the margin head of CAPAGENT-8B makes of those numbers. All values are from evaluation logs, the left boxes with the original tools and the right boxes with the swapped tools; coordinates are integers in $[0, 999]$ as in SpatialCLI.

H.1 One Call per Tool

Depth. DA-2K item 1203 shows a kitchen. Point 1 lies on the rim of a stainless-steel sink and point 2 on the wall tiles above the counter; point 1 is closer. The call carries a `depth_order` check, so $p = d_2 - d_1$, $\theta = 0$, and $s_\tau = (|d_1| + |d_2|)/2$ (Appendix E.4).

Original tools: Depth Anything 3

```
<tool_call>
{"name": "query_depth", "arguments": {
  "points": [[412, 655], [538, 301]],
  "image_indices": [1],
  "check": {"type": "depth_order",
    "a": "point_1", "b": "point_2"}}}
</tool_call>
<tool_response>
{"count": 2, "result": [
 {"point_2d": [412, 655], "depth": 1.42,
  "spread_m": 0.05, "valid_frac": 0.96,
  "edge": 0.31},
 {"point_2d": [538, 301], "depth": 1.71,
  "spread_m": 0.04, "valid_frac": 0.98,
  "edge": 0.12}]}
</tool_response>
[check depth(point_1) < depth(point_2):
 margins (+0.128, +0.243) -> USE]
[Tool-call budget: at most 9 more
 tool calls may be made.]
```

Swapped tools: Metric3D v2 Small

```
<tool_call>
{"name": "query_depth", "arguments": {
  "points": [[412, 655], [538, 301]],
  "image_indices": [1],
  "check": {"type": "depth_order",
    "a": "point_1", "b": "point_2"}}}
</tool_call>
<tool_response>
{"count": 2, "result": [
 {"point_2d": [412, 655], "depth": 1.63,
  "spread_m": 0.19, "valid_frac": 0.78,
  "edge": 0.31},
 {"point_2d": [538, 301], "depth": 1.74,
  "spread_m": 0.06, "valid_frac": 0.97,
  "edge": 0.12}]}
</tool_response>
[check depth(point_1) < depth(point_2):
 margins (-0.059, +0.190) -> RE-QUERY]
[Tool-call budget: at most 9 more
 tool calls may be made.]
```

Radius implied by the margins: 0.09 m with $s_\tau = 1.565$ m (left); 0.21 m with $s_\tau = 1.685$ m (right).

The `edge` values are the same on both sides because they are computed from the image. Metric3D v2 Small places the reflective sink rim 21 cm farther away and reports a spread almost four times larger on it, with a fifth of the pixels invalid. The order is still correct, by 11 cm instead of 29 cm, but the interval now crosses zero, so the loop asks for another point instead of accepting the order.

Pose. MMSI-Bench Pos-Cam-Cam item 214 shows a meeting room twice; the second photo is taken from farther left and closer to the table, with the camera turned right to keep the table in view. One call carries two checks, one per axis, and each gets its own verdict. Translation checks use $\theta = 0$ and $s_\tau = 0.1$ m, with p the translation in the named direction.

Original tools: VGGT

```
<tool_call>
{"name": "query_pose", "arguments": {
  "query": "camera motion",
  "image_indices": [1, 2],
  "check": [
   {"type": "camera_position",
    "axis": "x", "sign": "left"},
   {"type": "camera_position",
    "axis": "z", "sign": "front"}]}}
</tool_call>
<tool_response>
{"motion": {"yaw_deg": 14.8,
  "pitch_deg": -1.2, "roll_deg": 0.3,
  "t_m": [-0.46, 0.03, 0.12],
  "inlier_frac": 0.72, "reproj_px": 1.6,
  "overlap": 0.47}}
</tool_response>
[check position x left:
 margins (+3.50, +5.70) -> USE]
[check position z front:
 margins (+0.50, +1.90) -> USE]
[Tool-call budget: at most 9 more
 tool calls may be made.]
```

Swapped tools: MASt3R

```
<tool_call>
{"name": "query_pose", "arguments": {
  "query": "camera motion",
  "image_indices": [1, 2],
  "check": [
   {"type": "camera_position",
    "axis": "x", "sign": "left"},
   {"type": "camera_position",
    "axis": "z", "sign": "front"}]}}
</tool_call>
<tool_response>
{"motion": {"yaw_deg": 13.9,
  "pitch_deg": -0.8, "roll_deg": 0.5,
  "t_m": [-0.17, 0.02, 0.21],
  "inlier_frac": 0.44, "reproj_px": 3.2,
  "overlap": 0.47}}
</tool_response>
[check position x left:
 margins (-0.60, +4.00) -> RE-QUERY]
[check position z front:
 margins (+0.50, +3.70) -> USE]
[Tool-call budget: at most 9 more
 tool calls may be made.]
```

Radii implied by the margins: x 0.11 m and z 0.07 m (left); x 0.23 m and z 0.16 m (right).

The `overlap` field is identical because it comes from SIFT matching on the two images. MASt3R keeps fewer correspondences and has a larger residual, and its sideways estimate shrinks from 0.46 to 0.17 m. The head widens both intervals, but only the x interval crosses zero; the z check stays `USE`. A follow-up call therefore only has to settle left versus right, which is what the policy asks for next in this episode.

Locate. SpatialCLI-Bench item 305 asks whether an office has a blue recycling bin and where it is. Locate fuses Locate Anything (Wang et al., 2026b) with a detector; only the detector is swapped. The `present` check reads the presence margins of Appendix E.2, with $s_\tau = 1$.

Original tools: Locate Anything + Grounding DINO

```
<tool_call>
{"name": "query_locate", "arguments": {
  "query": "blue recycling bin",
  "image_indices": [1],
  "check": {"type": "present",
    "query": "blue recycling bin"}}}
</tool_call>
<tool_response>
{"count": 1, "result": [
 {"bbox_2d": [603, 388, 781, 702],
  "point_2d": [692, 545],
  "score": 0.62, "score_gap": 0.41,
  "box_frac": 0.056, "n_cand": 1,
  "edge": 0.19}]}
</tool_response>
[check present blue recycling bin:
 margins (+0.21, +0.43) -> USE]
[Tool-call budget: at most 9 more
 tool calls may be made.]
```

Swapped tools: Locate Anything + YOLO-World-L

```
<tool_call>
{"name": "query_locate", "arguments": {
  "query": "blue recycling bin",
  "image_indices": [1],
  "check": {"type": "present",
    "query": "blue recycling bin"}}}
</tool_call>
<tool_response>
{"count": 1, "result": [
 {"bbox_2d": [598, 391, 779, 699],
  "point_2d": [688, 545],
  "score": 0.31, "score_gap": 0.06,
  "box_frac": 0.055, "n_cand": 4,
  "edge": 0.19}]}
</tool_response>
[check present blue recycling bin:
 margins (-0.04, +0.30) -> RE-QUERY]
[Tool-call budget: at most 9 more
 tool calls may be made.]
```

Presence implied by the margins: 0.82 ± 0.11 (left); 0.63 ± 0.17 (right).

The two boxes almost coincide, but the raw scores do not: 0.31 would be a weak Grounding DINO detection and is an ordinary score for a correct YOLO-World box. The head does not read the score alone. The small gap to the second candidate (0.06) and the four candidates above 0.2, one of them a gray bin next to the blue one, are what widen the interval here.

Segment. Segment is backed by SAM 3 (Carion et al., 2026) in both settings, so a swap of the other three tools leaves it alone. The call below, from the same SpatialCLI-Bench question as the locate pair, returns byte-identical output in both settings and takes 1.2 s either way.

Original tools: SAM 3

```
<tool_call>
{"name": "query_segment", "arguments": {
  "query": "office chair",
  "image_indices": [1]}}
</tool_call>
<tool_response>
{"count": 2, "result": [
 {"bbox_2d": [118, 512, 344, 948],
  "point_2d": [231, 730],
  "polygon_2d": [[[152, 514], [301, 512],
    [344, 655], [322, 781], [338, 947],
    [131, 948], [118, 790], [140, 655]]]},
 {"bbox_2d": [402, 560, 571, 902],
  "point_2d": [486, 731],
  "polygon_2d": [[[431, 561], [548, 560],
    [571, 702], [555, 902], [402, 899],
    [411, 704]]]}]}
</tool_response>
[Tool-call budget: at most 8 more
 tool calls may be made.]
```

Swapped tools: SAM 3 (not replaced)

```
<tool_call>
{"name": "query_segment", "arguments": {
  "query": "office chair",
  "image_indices": [1]}}
</tool_call>
<tool_response>
{"count": 2, "result": [
 {"bbox_2d": [118, 512, 344, 948],
  "point_2d": [231, 730],
  "polygon_2d": [[[152, 514], [301, 512],
    [344, 655], [322, 781], [338, 947],
    [131, 948], [118, 790], [140, 655]]]},
 {"bbox_2d": [402, 560, 571, 902],
  "point_2d": [486, 731],
  "polygon_2d": [[[431, 561], [548, 560],
    [571, 702], [555, 902], [402, 899],
    [411, 704]]]}]}
</tool_response>
[Tool-call budget: at most 8 more
 tool calls may be made.]
```

Segment returns carry no quality fields and the policy attaches no check, so no verdict line is appended. We leave this tool exactly as SpatialCLI ships it.

H.2 Checks from Eight Evaluation Questions

Table 27 lists one checked call from each of eight evaluation questions with swapped tools. The margins follow from the returned values through the scales of Appendix E.4; the radius implied by each pair of margins is the half-width of the interval. Six rows end in a correct answer.

Table 27: **Checks from eight evaluation questions** (CAPAGENT-8B, swapped tools). Values are listed as point 1 / point 2 for depth. Yaw is positive to the right; checks on "left" use $p = -\text{yaw}$. Last column: final answer to the question.

Benchmark / item	Check	Key returned values	Margins	Verdict	Correct
SpatialCLI-Bench 041	depth_order: lamp $<$ sofa	d 2.08 / 2.64 m; spread 0.07 / 0.09; valid 0.95 / 0.96	(+0.161, +0.314)	`USE`	✓
SpatialCLI-Bench 203	depth_at_least: shelf $\geq$ 1.5 m	d 1.86 m; spread 0.05; valid 0.97; edge 0.14	(+0.160, +0.320)	`USE`	✓
MindCube 1182	camera_motion: yaw right	yaw +31.5°; inlier 0.58; reproj 2.2 px; overlap 0.41	(+2.37, +3.93)	`USE`	✓
MMSI Motion-Cam 086	camera_motion: yaw left	yaw $-6.3°$; inlier 0.29; reproj 5.1 px; overlap 0.49	(-0.08, +1.34)	`RE-QUERY` $\times 8$, then `UNUSABLE`	✗
MMSI Motion-Cam 209	camera_motion: z forward	t_z +0.42 m; inlier 0.66; reproj 1.9 px; overlap 0.57	(+2.70, +5.70)	`USE`	✓
MMSI Pos-Cam-Cam 133	camera_position: z behind	t_z -0.31 m; inlier 0.61; reproj 2.4 px; overlap 0.44	(+1.90, +4.30)	`USE`	✓
DA-2K 0877	depth_order: pt 1 $<$ pt 2	d 3.12 / 3.35 m; spread 0.06 / 0.05; valid 0.97 / 0.98; edge 0.09 / 0.08	(+0.028, +0.114)	`USE`	✗
DA-2K 1532	depth_order: pt 1 $<$ pt 2	d 0.94 / 1.21 m; spread 0.24 / 0.07; valid 0.62 / 0.95; second point: 0.97 / 1.22 m	(-0.130, +0.633), then (+0.137, +0.320)	`RE-QUERY`, then `USE`	✓

The two errors fail in opposite directions. On Motion-Cam item 086 the agent never accepts a small left turn read with an inlier fraction of 0.29, spends its budget, and then misreads the image motion on its own; this is the under-trust of Case 4 in Appendix L. On DA-2K item 0877 it accepts a reading it should not: Metric3D v2 Small returns a low spread on a flat, distant wall with few edges, which is usually the signature of a good reading, and the head predicts a radius of 0.14 m. The calibrated reference radius for this output is 0.29 m, which would have crossed zero, and the true order is the opposite. Nothing in the packet separates this reading from a reliable one (Table 23). Item 1532 shows the ordinary repair: a point on a glass pane gives a wide interval, and a second point on the window frame settles the order.

H.3 A COMPLETE EPISODE

Box H.1 reproduces one full episode from the swapped-tools evaluation, including every turn the agent writes, every tool return, the verdict lines, and the budget notices, followed by the token and time accounting from the log. It shows the case that motivates the image cue: the plant's depth reading has a spread more than twice the printer's, but the plant's leaves also give a high edge energy, so the head reads most of that spread as scene structure and accepts the order in one call.

Box H.1: Full episode, SpatialCLI-Bench ID 452, CAPAGENT-8B, swapped tools

```
[system]  SpatialCLI agentic prompt + quality-field and check additions
          (1,412 tokens, not shown)
[user]    <image 1: office corner, 1280 x 960>
          Which is farther from the camera, the potted plant or the printer?
          Is the farther object to the left or to the right of the desk lamp?
          A: plant, left of lamp      B: plant, right of lamp
          C: printer, left of lamp    D: printer, right of lamp
          E: same distance; plant left    F: same distance; printer right
          Answer with {"answer": "<letter>"}.

[assistant, turn 1]
<think>
I need two things: which of the plant and the printer is farther, and whether that
object is left or right of the lamp. First I will locate all three objects.
```

```
</think>
<tool_call>
{"name": "query_locate", "arguments": {"query": "potted plant", "image_indices": [1],
 "check": {"type": "present", "query": "potted plant"}}}
</tool_call>
<tool_call>
{"name": "query_locate", "arguments": {"query": "printer", "image_indices": [1],
 "check": {"type": "present", "query": "printer"}}}
</tool_call>
<tool_call>
{"name": "query_locate", "arguments": {"query": "desk lamp", "image_indices": [1],
 "check": {"type": "present", "query": "desk lamp"}}}
</tool_call>
<tool_response>
{"count": 1, "result": [{"bbox_2d": [96, 402, 268, 811], "point_2d": [182, 606],
  "score": 0.34, "score_gap": 0.22, "box_frac": 0.070, "n_cand": 1, "edge": 0.41}]}
{"count": 1, "result": [{"bbox_2d": [702, 455, 889, 612], "point_2d": [795, 533],
  "score": 0.29, "score_gap": 0.19, "box_frac": 0.029, "n_cand": 2, "edge": 0.22}]}
{"count": 1, "result": [{"bbox_2d": [455, 268, 541, 521], "point_2d": [498, 394],
  "score": 0.27, "score_gap": 0.15, "box_frac": 0.022, "n_cand": 2, "edge": 0.30}]}
</tool_response>
[check present potted plant: margins (+0.17, +0.41) -> USE]
[check present printer: margins (+0.11, +0.37) -> USE]
[check present desk lamp: margins (+0.06, +0.34) -> USE]
[Tool-call budget: at most 7 more tool calls may be made.]

[assistant, turn 2]
<think>
All three are found. Now the depth at the plant and printer centers, with a check
that the plant is closer than the printer.
</think>
<tool_call>
{"name": "query_depth", "arguments": {"points": [[182, 606], [795, 533]],
 "image_indices": [1], "check": {"type": "depth_order", "a": "point_1", "b": "point_2"}}}
</tool_call>
<tool_response>
{"count": 2, "result": [
 {"point_2d": [182, 606], "depth": 2.31, "spread_m": 0.13, "valid_frac": 0.84,
  "edge": 0.41},
 {"point_2d": [795, 533], "depth": 2.87, "spread_m": 0.05, "valid_frac": 0.97,
  "edge": 0.22}]}
</tool_response>
[check depth(point_1) < depth(point_2): margins (+0.135, +0.297) -> USE]
[Tool-call budget: at most 6 more tool calls may be made.]

[assistant, turn 3]
<think>
The check is decided: the plant is at 2.31 m and the printer at 2.87 m, so the printer
is farther. The plant reading has a higher spread, but the point is on the leaves,
which have many edges, and the check still holds. Printer x = 795 and lamp x = 498,
so the printer is to the right of the lamp. Option D.
</think>
{"answer": "D"}

# log summary
# result: correct (D). executed calls: 4 (3 locate, 1 depth). checks: 4 USE, 0 RE-QUERY
# tokens: prompt 2,114 (system 1,412 + image 614 + question 88)
#         generated 327 (turn 1: 142, turn 2: 64, turn 3: 121); tool returns 402
# wall-clock (s): turn 1 6.9 | locate x3 8.1 | margin head x3 0.63 | turn 2 4.4
#                 depth 3.7 | margin head 0.21 | turn 3 8.1 | loop overhead 1.1
#                 total 33.1
```

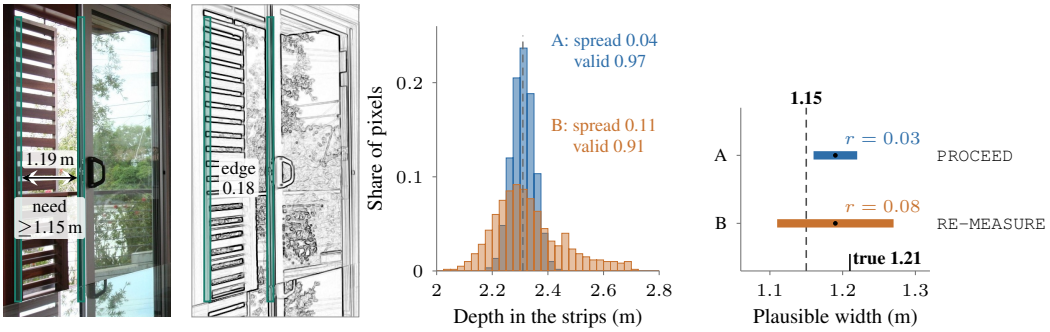
The margins of the depth check correspond to $p = 0.56\,\text{m}$, $s_\tau = 2.59\,\text{m}$, and a radius of 0.21 m. The episode takes 33.1 s, close to the 35.0 s average for CAPAGENT-8B in Table 26; the tool calls take 11.8 s of it, and the four margin-head readings 0.84 s.

I VISUAL EXAMPLES

This appendix shows three things that the tables summarize: how one packet is built from a frame and a provider's depth map, how spread relates to error for individual queries and pairs, and what the checks do inside SpatialCLI's loop.

I.1 How a Depth Packet Is Built

Figure 6 follows the doorway of Case 1 (Appendix L) from the frame to the action. The clearance query is the union of two vertical strips, one on each side of the opening (panel a). The strips are 20 pixels wide in the 840×1332 crop shown there and cover the part of the opening between the floor and the top of the door frame. Edge energy is computed from the image alone, so panel (b) is the same for every provider: inside the strips it is 0.18, below the median of 0.27 over the reference provider's core depth queries (Figure 7a), because the strips lie mostly on the smooth aluminium frame and the glass beside it. Panel (c) shows 600 depth pixels sampled inside the strips from each provider's output. Both providers put the median at 2.31 m, and back-projecting the two strip edges with these medians gives the same clearance, 1.19 m. They differ in the shape of the distribution. UniDepth's pixels are tight (spread 0.04 m) and 18 of them are invalid. Metric3D v2 Small spreads the frame over a wider range and adds a tail between 2.45 and 2.70 m from pixels that read through or reflect off the glass; its spread is 0.11 m and 54 pixels are invalid. After calibration these become radii of 0.03 and 0.08 m, and only the second interval crosses the requirement (panel d). The agent never sees panels (c) and (d). It sees the four packet fields; depth and edge are the same in the two packets, and the two that change, spread and valid fraction, are the ones a model swap moves.



(a) Query strips (b) Image edges (c) Per-pixel depth (d) Interval and action

Figure 6: **How the Case 1 depth packet is built.** (a) RGB frame with the two query strips and the measured opening. (b) Sobel edge magnitude of the grayscale frame (kernel divided by 4, so a unit intensity step has magnitude 1; computed at 420-pixel width; darker is stronger); the mean inside the strips is the packet's `edge`. (c) Depth of the pixels inside the strips for provider A (UniDepth, blue) and provider B (Metric3D v2 Small, orange); the dashed line is the shared median. (d) Calibrated plausible width against the 1.15 m requirement, the reference actions, and the true width.

I.2 Spread Across Scenes and Within Pairs

Figure 7 shows the slopes of Section 5.5 at the level of single queries. Panel (a) plots the reference radius against spread for 150 core depth queries of the reference provider, drawn at random and including Case 1. Queries with many image edges (triangles) have larger spread, often 0.08–0.15 m, but their radius is not much larger than that of edge-poor queries: much of their spread sits on door frames, shelves, and furniture boundaries inside the region, where the depth is right. The least-squares slope is 0.21. Panel (b) plots the change in reference radius against the change in spread for 400 same-scene pairs, 100 per depth treatment. Here the scene and query are fixed and only the provider changes, so a rise in spread goes with the provider, including how that provider reacts to this particular scene; the slope is 0.63. The pairs of the noisiest replacement (Metric3D v2 Small, $3.41\times$) lie furthest out, but they follow the same line as the others; the relation between a change in spread and a change in error is shared across treatments, which is what lets an agent trained on one depth treatment (ZoeDepth-NK) transfer to the other three. The single-output margin policy uses nearly one response for both kinds of change (0.27 across scenes and 0.30 within pairs), so in panel (b) it predicts less than half of the change that a swap causes. CAPAGENT separates them (0.23 and 0.57). Both are the controlled indoor policies of Table 2. The Case 1 pair (star, $+0.07$ m of spread and $+0.05$ m of radius) lies on the within-pair line. For its packet B, the single-output margin policy predicted a radius of 0.035 m, just short of the 0.04 m at which the interval reaches 1.15 m, while CAPAGENT predicted 0.075 m.

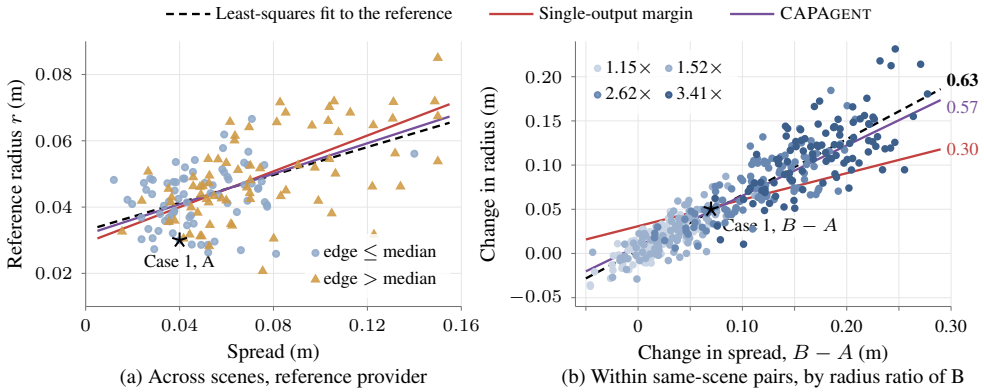


Figure 7: **A rise in spread means different things across scenes and within pairs.** (a) Reference radius against spread for 150 core depth queries of the reference provider (UniDepth), split at the median edge energy (0.27). (b) Change in reference radius against change in spread for 400 same-scene depth pairs, shaded by the radius ratio of the replacement. Dashed: least-squares fit to the plotted points (0.21 and 0.63, as for the full core in Section 5.5; panel b uses one intercept per treatment). Solid: slopes of the controlled single-output margin policy and CAPAGENT measured on the full core, drawn through the centroid of the points. Stars: the Case 1 packet and pair.

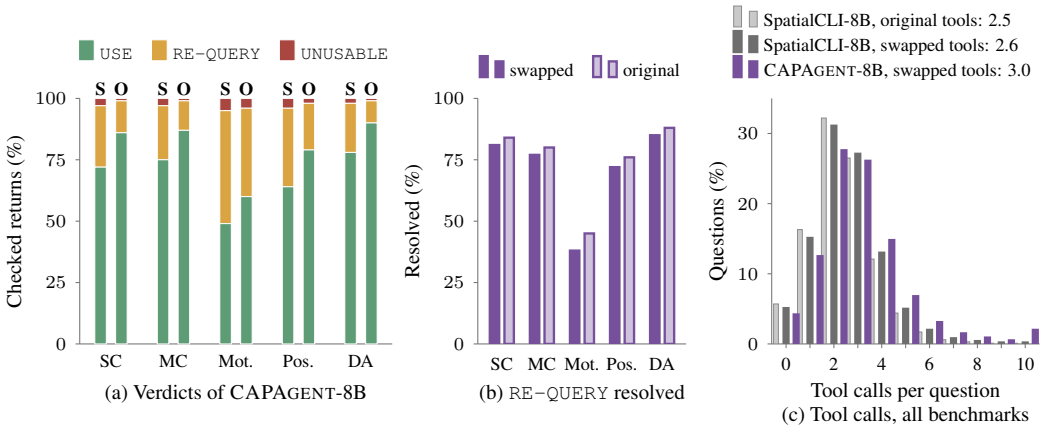
I.3 What the Agent Does with Its Checks

Figure 8 summarizes the verdict lines of CAPAGENT-8B in SpatialCLI's loop. With the original tools, most checked returns are used directly (86–90% on SpatialCLI-Bench, MindCube, and DA-2K). After the swap, `RE-QUERY` becomes more common on every benchmark, and most on the two MMSI subsets, where MASt3R replaces VGGT: 46% of checked returns on camera motion and 32% on camera position (panel a). `UNUSABLE` stays at 1–5% of returns. Panel (b) asks whether a re-query pays off, that is, whether the same check later ends in `USE` before the budget runs out. It does for 72–85% of re-queries on four benchmarks and for only 38% on camera motion (Case 4, Appendix L). Panel (c) shows where the extra calls go. SpatialCLI-8B makes 2.5 calls per question with the original tools and 2.6 with swapped tools; CAPAGENT-8B makes 3.0 with swapped tools. The difference sits at four to six calls and in the 2.0% of questions that reach the ten-call budget (0.2% for SpatialCLI-8B). On SpatialCLI-Bench, 64% of the questions that CAPAGENT-8B answers correctly and SpatialCLI-8B answers wrongly contain at least one `RE-QUERY`, so most of the answers it gains go through a re-query.

J Scene Gallery

Figure 9 shows the kinds of scenes behind our tasks and behind the standard benchmarks of Table 1. The indoor frames of CAPSWAP-BENCH come from ScanNet and ScanNet++ (Dai et al., 2017; Yeshwanth et al., 2023), whose terms of use allow figures in papers but not redistribution of the images, so the first row draws the three indoor tasks on public images from DA-2K and MMSI-Bench (Yang et al., 2024; 2026); these panels show what each task measures and are not scored items of CAPSWAP-BENCH. The other panels are items as released by the public benchmarks, and all overlays are drawn by us. Cases 2 and 4 follow panels (e) and (h) through the loop, and Case 3 follows the two frames of panel (b) as the MMSI-Bench camera-position item they come from.

Row 1: the three indoor tasks. In (a), the passage query asks whether the aisle between two shelf units is at least 1.15 m wide. The depth tool reads the provider's depth inside the two strips, back-projects their inner edges, and returns the clearance together with the spread and valid fraction of each strip. Book spines make the strips busy, so part of the spread here comes from real edges in the scene, which is the situation Proposition 1 describes. In (b), the cache-reuse query compares a cached view with the current view of the same lounge. The pose tool returns the translation and rotation between them, and the task rule checks both against the tolerances θ_t and θ_R; the agent derives θ_t from the scene range in the depth packet. The white chairs and dark carpet give few stable

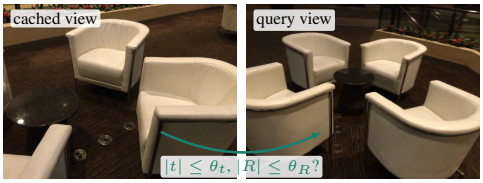


(a) Verdicts of CAPAGENT-8B (b) RE−QUERY resolved (c) Tool calls, all benchmarks

Figure 8: **Checks inside SpatialCLI's loop.** (a) Verdicts of CAPAGENT-8B per checked tool return, with swapped (S) and original (O) tools. (b) Share of RE−QUERY verdicts after which the same check ends in USE within the budget. (c) Distribution of executed tool calls per question, pooled over the five benchmarks of Table 1; means in the legend. SC: SpatialCLI-Bench; MC: MindCube; Mot./Pos.: MMSI camera motion and camera position; DA: DA-2K.



(a) *Passage*: is the aisle between the shelves wide enough? DA-2K (indoor).



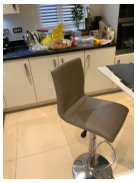
(b) *Cache reuse*: can the cached view stand in for the query view? MMSI-Bench, item 495.



(c) *Object selection*: three candidate boxes for one request. DA-2K (indoor).



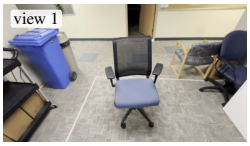
(d) *Outdoor distance*: is the lane ahead free for the next 8 m? KITTI.



(e) SpatialCLI-Bench: kettle vs. bar stool.

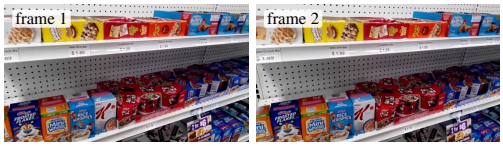


(f) DA-2K (transparent): is point 1 or point 2 closer? Answer: 2.





(g) MindCube: from view 1 to view 2, which way did I move? Answer: diagonally forward and left.



(h) MMSI-Bench Motion-Cam, item 213: which way is the camera rotating? Answer: left.

Figure 9: **Representative scenes.** Row 1: the three indoor tasks of CAPSWAP-BENCH, drawn on public images of similar scenes. Row 2: the outdoor distance task and two single-image benchmark items. Row 3: two-image benchmark items. Image sources and licenses: DA-2K (a, c, f; Apache-2.0 dataset release; Yang et al., 2024), MMSI-Bench (b, h; CC BY 4.0; Yang et al., 2026), KITTI (d; CC BY-NC-SA 3.0; Geiger et al., 2012), SpatialCLI-Bench (e; image from CA-1M, CC BY-NC-ND 4.0, shown without overlay; Zhou et al., 2026; Lazarow et al., 2025), and MindCube (g; MIT; Wang et al., 2026a). All strips, boxes, points, arrows, and text tags are drawn by us. Panel (e) carries none.

correspondences, so the inlier fraction of such pairs is often low even when the pose is right. In (c), the request names an object by attribute and place. Three cushions share the same pattern, so a detector returns close scores for all three, and the gap between the top two scores is small. A small gap pushes the grounding margins toward RE−LOOK, even when the top box is the right one.

Row 2: outdoor distance and single-image benchmarks. In (d), the outdoor query asks whether the lane ahead is free for the next 8 m. The depth tool reads the region between the parked cars; ground truth comes from the projected LiDAR points. Panel (e) is a SpatialCLI-Bench item that needs three tools: locate finds the kettle and the bar stool, depth orders them, and pose reports which side of the kettle faces the camera. The answer is that the kettle is higher in the image, further from the camera, and shows its left side. In (f), a DA-2K pair asks whether a point on the armchair or a point on the water glass is closer; the glass is. Depth models are least reliable on transparent and reflective surfaces, and DA-2K keeps such scenes as a separate category for that reason.

Row 3: two-image benchmarks. Panel (g) is a MindCube item: two views of an office corner, and the question is how the camera moved between them (diagonally forward and left). Panel (h) is an MMSI-Bench Motion-Cam item: the camera turns slightly to the left along a supermarket shelf. Both are decided by the pose tool. Items like (h) combine a small rotation with repeated packaging, which makes correspondences ambiguous and lowers the inlier fraction even when the two frames overlap almost completely. Case 4 follows this item through the loop; it is where CAPAGENT-8B keeps asking for another reading.

K PROMPT TEMPLATES

This section lists the prompts used for training and evaluation. Text in braces is filled in by the data loader. The prompts are identical for every provider, so after a swap the only thing that changes in the agent's input is the numbers.

CAPSWAP-BENCH decisions. Box K.1 is the prompt for all CAPSWAP-BENCH decisions. The system message says that the backing model can change and is not named, which matches the deployment protocol. Packet fields are written as a flat list with units instead of JSON; each field has a short description so that the same prompt serves the three tasks. The reply starts with the three anchor tokens. They come before any explanation, so the hidden states that the margin head reads do not depend on how long the explanation is. An indoor record, in training and in CAPSWAP-BENCH, shows one depth query and one pose query of the same capture unit (Appendix C.1). The reason and action lines answer the record's task; the second query is read only through its anchor, where L_{state} fits its reference margins and the retain term keeps them in place under a swap. An anchor whose capability has no query in the record, such as <obj> here, is still emitted, and its coordinates are masked in L_{state}.

Box K.1: CAPSWAP-BENCH decision prompt

```
[system]
You are the decision module of a mobile robot. A perception tool has measured
part of the scene for you. The tool may be backed by different models at
different times, and you are not told which model produced this reading.
Read the measurement together with its quality fields, decide how far the
evidence is from the task limit, and choose one allowed action.
Begin your reply with the tokens <depth> <pose> <obj> in this order.
Then write one line starting with "Reason:" and one line starting with "Action:".

[user]
Task: passage
Question: Can the robot pass through the opening in the query region?
Requirement: clear width >= {theta} m
Query region: {box_2d}   (x1, y1, x2, y2 in 0-999 image coordinates)
Allowed actions: PROCEED | STOP | RE-MEASURE

Depth packet (for this task):
  depth_m    = {p}        measured clear width, in metres
  spread_m   = {spread}   robust spread of depth in the region (IQR / 1.349), in metres
  valid_frac = {valid}    fraction of region pixels with a valid depth value
  edge       = {edge}     image edge energy of the region, computed from the RGB image
  range_m    = {range}    median depth over the whole view, in metres
Pose packet (second query of this unit; no action is asked for it):
  entry {i}: t_m = {p_t}  rot_deg = {p_R}  inlier_frac = {inl}
             reproj_px = {res}  overlap = {ov}
Grounding packet: none in this record

# cache_reuse records swap the two roles: the pose packet belongs to the task, with one
#   block per cache entry, and the depth packet is the second query. The task lines give
```

```
#   Tolerance: rotation <= {theta_R} deg; translation <= range_m x tan({2*scale} deg)
#   Allowed actions: USE-CACHE | REFRESH | RE-MEASURE
# object_selection records hold one grounding block for the top box and no other packet:
#   score = {s}  score_gap = {g}  box_frac = {b}  n_cand = {n}  edge = {e}
#   Allowed actions: ACT | ABSENT | RE-LOOK
```

Target responses. Targets are produced by a fixed template from the reference margins (Box K.2), so the reason line can never disagree with the target action. The reason reports the plausible range $[p - r,\ p + r]$ rather than the radius r, because the radius is not something the agent can observe. The cue clause names the one to three fields that lie furthest from their medians over the training packets of the same task, measured in units of each field's interquartile range. When the scene annotation marks a structure in the query region, such as a door frame, a window, or a shelf edge, the clause names it too. We keep the reason short because longer explanations made the response slower without changing the head's margins.

Box K.2: Target responses for CAPSWAP-BENCH

```
# reference action PROCEED (packet A in Figure 1)
<depth> <pose> <obj>
Reason: The width reads 1.19 m with a low spread (0.04 m) and 97% valid pixels,
so the plausible width is about 1.16–1.22 m, above the required 1.15 m.
Action: PROCEED

# reference action RE-MEASURE (packet B in Figure 1)
<depth> <pose> <obj>
Reason: The width reads 1.19 m, but the spread is high (0.11 m) for a region with
few image edges (edge 0.18), and 9% of pixels are invalid. The plausible width is
about 1.11–1.27 m, which crosses the required 1.15 m.
Action: RE-MEASURE

# template, filled from the reference margins
Reason: The {quantity} reads {p} {unit}{cue_clause}. The plausible {quantity} is
about {p - r}–{p + r} {unit}, which {is above | is below | crosses} the required
{theta} {unit}.
Action: {q(phi)}
```

Additions to the SpatialCLI prompt. Single-output-8B, GAR-8B, and CAPAGENT-8B use SpatialCLI's agentic prompt with the text in Box K.3 appended. Tool-use SFT-8B sees the same quality fields and `check` argument but has no margin head, so it receives no verdict lines and decides in text. The SpatialCLI rows in Table 1 use SpatialCLI's original prompt and tool returns. The verdict line is not produced by the tool. After each depth, pose, or locate return, the policy emits its anchor tokens, the margin head reads their hidden states, and the loop appends the verdict to the observation before the next assistant turn. The budget notices are SpatialCLI's own wording.

Box K.3: Text appended to SpatialCLI's agentic prompt

```
Quality fields. Every query_depth, query_pose, and query_locate result also
carries quality fields for that reading:
  query_depth:  spread_m, valid_frac, edge
  query_pose:   inlier_frac, reproj_px, overlap
  query_locate: score, score_gap, box_frac, n_cand, edge
The model behind each tool can change between deployments and is not named.

Checks. When you call query_depth, query_pose, or query_locate, add a "check"
argument that states what you want to decide with the reading, for example
  {"type": "depth_order", "a": "point_1", "b": "point_2"}      is a closer than b?
  {"type": "camera_motion", "axis": "yaw", "sign": "left"}     did the camera turn left?
  {"type": "camera_position", "axis": "z", "sign": "front"}    is camera 2 in front?
  {"type": "present", "query": "red mug"}                      is the object there?
Several checks on one reading can be given as a list.
After the result you will see one line
  [check ...: margins (lo, hi) -> VERDICT]
  USE       the reading decides the check. Rely on it.
  RE-QUERY  the reading does not decide the check. If budget remains, query again
            with a tighter crop, another point on the same object, or the two
            images in the other order.
  UNUSABLE  no budget is left and the check is still undecided. Do not use this
            reading; answer from the rest of your evidence.

```

```
Budget notices (unchanged from SpatialCLI):
[Tool-call budget: at most N more tool calls may be made.]
[Tool-call budget exhausted: no more tool calls are allowed. Answer the question
immediately using the available information.]
```

Check-format training examples. The 2,000 check-format examples are written by a script from the scene annotations of training scenes (Box K.4); no language model writes them. They contain 800 depth ordering, 500 signed camera motion, 400 object presence, and 300 localization examples. The tool returns are produced live by the training providers on the training scenes, so the cue values in these examples come from the same models as the rest of the training data. About a third of the examples contain a `RE-QUERY` step, which is where the policy learns the follow-up strategies listed in Box K.3.

Box K.4: Generator for check-format examples (depth ordering shown)

```
# Written by a script from scene annotations of training scenes.
# Tool returns come from the training providers only. No language model
# writes these examples.

question = "Which is closer to the camera, the {name_a} or the {name_b}?"
call_1   = query_depth(points=[{pt_a}, {pt_b}],
                       check={"type": "depth_order", "a": "point_1", "b": "point_2"})
return_1 = provider output with cue fields, run on the training scene
target_1 = "<depth> <pose> <obj>" + verdict from the reference margins
if verdict_1 == "RE-QUERY":
    # pt_a2, pt_b2: pixels of the same instance masks with the smallest
    # reference radius, at least 25 px away from pt_a and pt_b
    call_2   = query_depth(points=[{pt_a2}, {pt_b2}], check=...)
    return_2 = provider output with cue fields
    target_2 = "<depth> <pose> <obj>" + verdict from the reference margins
think    = sentence templates, for example
           "The check is not decided; the reading on the {name} has a wide
            spread. I will query another point on the {name}."
answer   = "{name_closer} is closer", from the ground-truth depth
```

Prompt for frontier models. GPT-5.6 Sol and Qwen3.5-397B-A17B receive SpatialCLI's agentic prompt, the same four tools, and tool returns with the quality fields, together with the field descriptions in Box K.5. They do not receive verdict lines, since those come from our margin head.

Box K.5: Field descriptions given to frontier models

```
{SpatialCLI agentic tool-use prompt, unchanged}

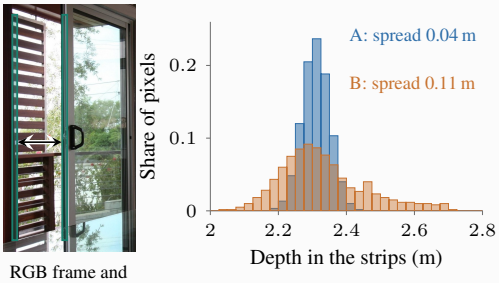
Quality fields. Tool results include extra fields that describe how reliable each
reading is:
  spread_m     robust spread of depth values around the queried point, in metres
  valid_frac   fraction of pixels around the point with a valid depth value
  edge         image edge energy around the point, computed from the image
  inlier_frac  fraction of image correspondences consistent with the pose
  reproj_px    median reprojection error of those correspondences, in pixels
  overlap      fraction of the image area visible in both images
  score        detector confidence for the returned box
  score_gap    gap between the best and second-best box scores
  box_frac     box area as a fraction of the image
  n_cand       number of candidate boxes with score above 0.2
The model behind each tool is not named and may differ from the one you are used
to. The same field value can mean a different error size for a different model.
Use these fields to decide whether a reading is reliable enough for your answer,
or whether to query again.
```

L CASE STUDIES

Case 1 is a CAPSWAP-BENCH replay episode with the controlled indoor policies of Table 2; Cases 2 to 4 come from the standard benchmarks with swapped tools and the 8B agents of Table 1 (Section 5.2), with traces copied from the evaluation logs and shortened where marked *[. . .]*. Green and red boxes end in correct and wrong answers. Purple lines are verdicts appended after the margin head reads a return; gray lines are SpatialCLI's budget notices.

L.1 CASE 1: A DOORWAY WITH AN UNSEEN DEPTH MODEL

Box L.1: Task (CAPSWAP-BENCH replay, indoor evaluation split, episode 214)



RGB frame and query strips

Left: RGB frame with the two query strips on the sides of the opening. Right: per-pixel depth inside the strips from provider A (UniDepth, training reference) and provider B (Metric3D v2 Small, unseen). Figure 6 shows the full construction of this packet.

The robot faces an open glass sliding door. It needs a clear width of at least 1.15 m. Allowed actions: `PROCEED`, `STOP`, `RE-MEASURE`. Both packets come from the same frame and query region; the agent sees one of them, without the provider's name.

Packet	`depth_m`	`spread_m`	`valid_frac`	`edge`	Ref. radius	Ref. action
A (UniDepth)	1.19	0.04	0.97	0.18	0.03 m	`PROCEED`
B (Metric3D v2 S)	1.19	0.11	0.91	0.18	0.08 m	`RE-MEASURE`

Ground truth. Clear width from the reconstructed mesh: 1.21 m. At the next keyframe, 0.42 m closer, provider B's reading has reference radius 0.025 m and reference action `PROCEED`.

Single-output margin policy, packet B (unsupported)

Step 1. Input: packet B.
<depth> <pose> <obj>
Reason: The width reads 1.19 m. The spread (0.11 m) is higher than usual; this looks like the edge of the door frame rather than noise in the reading. The plausible width is about 1.16–1.22 m, above the required 1.15 m.
Action: PROCEED
[head margins (+0.004, +0.065) –> PROCEED; predicted radius 0.035 m]
Episode ends: the robot drives through.
The door is 1.21 m wide, so the robot passes. The commitment was not supported: with the calibrated radius of 0.08 m, the reading allows widths down to 1.11 m.

CAPAGENT, packet B (correct)

Step 1. Input: packet B.
<depth> <pose> <obj>
Reason: The width reads 1.19 m, but the spread is high (0.11 m) for a region with few image edges (edge 0.18), and 9% of pixels are invalid, likely on the glass. The plausible width is about 1.12–1.27 m, which crosses the required 1.15 m.
Action: RE-MEASURE
[head margins (–0.030, +0.100) –> RE-MEASURE; predicted radius 0.075 m]
Step 2. Input: provider B at the next keyframe
depth_m = 1.18, spread_m = 0.05, valid_frac = 0.96, edge = 0.21
<depth> <pose> <obj>
Reason: The width reads 1.18 m with a low spread (0.05 m) and 96% valid pixels. The plausible width is about 1.16–1.20 m, above the required 1.15 m.
Action: PROCEED
[head margins (+0.009, +0.043) –> PROCEED; predicted radius 0.020 m]

Analysis. Both agents read the same width and the same `edge`, which comes from the image and is therefore identical in the two packets. The single-output margin policy turns a spread of 0.11 m into a radius of 3.5 cm, about what such a spread meant for scene edges on its training providers. CAPAGENT turns it into 7.5 cm, close to the calibrated 8 cm, because on a region with few edges the extra spread more likely comes from the model. The single-output policy was right about this

door, but its evidence did not rule out 1.11 m: on Metric3D v2 Small packets whose reference action is `RE-MEASURE`, commitments like this one are wrong about the opening in 23% of episodes.

L.2 Case 2: Two Near-Tied Detector Boxes

Box L.2: Task (SpatialCLI-Bench, RefSpatial item c660de07, swapped tools)

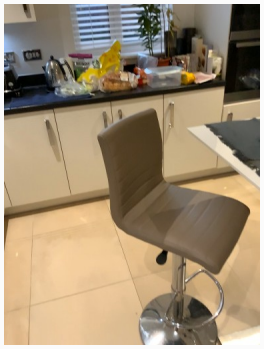


Image 1 (shown as released, resized). A kitchen seen from beside a bar stool. The stool fills the lower half of the frame. On the far counter stand a toaster, cut off by the left edge, a silver electric kettle, a yellow bag, and food containers.

Consider the silver electric kettle and the beige bar stool. Which of these two objects is positioned higher up in the image, is that higher object closer to or further away from the camera than the other, and which of its sides is visible?
Options:
A: The beige bar stool, closer, its right side is visible
B: The silver kettle, further away, its left side is visible
C: The silver kettle, further away, its right side is visible
D: The silver kettle, closer, its left side is visible
E: The beige bar stool, further away, its left side is visible
F: The beige bar stool, closer, its left side is visible
[SpatialCLI's instruction to verify every clause and to answer with `{"answer": "C"}` follows.]

Ground truth. Correct option: **B**. Tools: Locate runs Locate Anything with YOLO-World-L and Depth runs Metric3D v2 Small, both unseen by the adapted policies; object orientation stays with Orient Anything V2 (Wang et al., 2025c).

SpatialCLI-8B, swapped tools (incorrect)

<think>
Three clauses: which object is higher in the image, whether it is closer or further, and which side is visible. Plan: locate both objects, compare their depth, then ask for the orientation of the higher one.
</think>
<tool_call>
{"name": "query_locate", "arguments": {"query": "silver electric kettle", "image_indices": [1]}}
</tool_call>
<tool_response>
{"count": 1, "result": [{"bbox_2d": [0, 170, 65, 264], "point_2d": [33, 217]}]}
</tool_response>
[Tool–call budget: at most 9 more tool calls may be made.]
[query_locate for "beige bar stool": box [362, 300, 962, 999], point [650, 585]]
[Tool–call budget: at most 8 more tool calls may be made.]
<tool_call>
{"name": "query_depth", "arguments": {"points": [[33, 217], [650, 585]], "image_indices": [1]}}
</tool_call>
<tool_response>
{"count": 2, "result": [{"point_2d": [33, 217], "depth": 2.52}, {"point_2d": [650, 585], "depth": 1.08}]}
</tool_response>
[Tool–call budget: at most 7 more tool calls may be made.]
<think>
The kettle is higher (y = 217 against 585) and further away (2.52 m against 1.08 m). Now its visible side.
</think>
<tool_call>
{"name": "query_pose", "arguments": {"query": "object orientation", "bbox_2d": [0, 170, 65, 264], "image_indices": [1]}}
</tool_call>
<tool_response>
{"orientation": {"visible_side": "right", "confidence": 0.64}}
</tool_response>
[Tool–call budget: at most 6 more tool calls may be made.]
<think>
The kettle is higher, further away, and its right side is visible. Option C.
</think>
{"answer": "C"}

CAPAGENT-8B, swapped tools (correct)

<think>
Three clauses: which object is higher, closer or further, and which side is visible. First I need both objects, with a check that each one is found.
</think>
<tool_call>
{"name": "query_locate", "arguments": {"query": "silver electric kettle", "image_indices": [1], "check": {"type": "present", "query": "silver electric kettle"}}}
</tool_call>
<tool_response>
{"count": 1, "result": [{"bbox_2d": [0, 170, 65, 264], "point_2d": [33, 217], "score": 0.27, "score_gap": 0.03, "box_frac": 0.006, "n_cand": 2, "edge": 0.37}]}
</tool_response>
[check present silver electric kettle: margins (–0.09, +0.27) –> RE–QUERY]
[Tool–call budget: at most 9 more tool calls may be made.]
<think>
The check is not decided. There are two candidates with almost the same score (gap 0.03), and the top box touches the left edge of the image, where only part of an appliance is visible. I will describe the kettle more precisely.
</think>
<tool_call>
{"name": "query_locate", "arguments": {"query": "silver electric kettle with a black handle, next to the yellow bag", "image_indices": [1], "check": {"type": "present", "query": "silver electric kettle"}}}
</tool_call>
<tool_response>
{"count": 1, "result": [{"bbox_2d": [158, 156, 254, 252], "point_2d": [206, 204], "score": 0.41, "score_gap": 0.24, "box_frac": 0.009, "n_cand": 1, "edge": 0.29}]}
</tool_response>
[check present silver electric kettle: margins (+0.12, +0.38) –> USE]
[Tool–call budget: at most 8 more tool calls may be made.]
[query_locate for "beige bar stool" with a present check: box [362, 300, 962, 999], point [650, 585], score 0.52, score_gap 0.44, verdict USE]
[Tool–call budget: at most 7 more tool calls may be made.]
<tool_call>
{"name": "query_depth", "arguments": {"points": [[650, 585], [206, 204]], "image_indices": [1], "check": {"type": "depth_order", "a": "point_1", "b": "point_2"}}}
</tool_call>
<tool_response>
{"count": 2, "result": [{"point_2d": [650, 585], "depth": 1.08, "spread_m": 0.05, "valid_frac": 0.98, "edge": 0.12}, {"point_2d": [206, 204], "depth": 2.46, "spread_m": 0.13, "valid_frac": 0.88, "edge": 0.29}]}
</tool_response>
[check depth(point_1) < depth(point_2): margins (+0.62, +0.94) –> USE]
[Tool–call budget: at most 6 more tool calls may be made.]
[query_pose with "object orientation" on box [158, 156, 254, 252]: visible_side "left", confidence 0.71]
[Tool–call budget: at most 5 more tool calls may be made.]
<think>
The kettle is higher in the image (y = 204 against 585), further away (2.46 m against 1.08 m, decided), and its left side is visible. Option B.
</think>
{"answer": "B"}

Analysis. With Grounding DINO behind the locate tool, the same first call returns the kettle as the only candidate (score 0.58, gap 0.33; we reran it with the original tools). YOLO-World-L puts the toaster at the left edge 0.03 above the kettle. SpatialCLI-8B takes the top box, which is what its training rewarded, and everything after that is correct for the toaster: it is higher than the stool, further away, and shows its right side. The depth clause was never in doubt; the wrong answer comes from the object. CAPAGENT-8B's verdict does not come from the raw score. On YOLO-World-L, a score near 0.3 is ordinary for a correct box (Appendix H); the small gap and the second candidate are what leave the presence check undecided.

L.3 Case 3: Camera Position with a Swapped Pose Model

Box L.3: Task (MMSI-Bench, Pos-Cam-Cam, item 495, swapped tools)



Image 1



Image 2

A lounge with white leather tub chairs on a dark striped carpet. Image 1 is a close view of two chairs and a round black table, with a planter along the far edge. Image 2 shows four chairs around a small black table from the side, with the planter at the upper right. The same pair appears in Figure 9(b).

In the same scene, from which direction relative to the first photo was the second photo taken?
Options: A: Rear right, B: Front left, C: Rear left, D: Front right

Ground truth. Correct option: **D**. Tools: camera motion is backed by MASt3R instead of VGGT.

SpatialCLI-8B, swapped tools (incorrect)

<think>
The question asks where camera 2 is relative to camera 1: left or right, and front or rear. I can get this from the camera motion between the two images.
</think>
<tool_call>
{"name": "query_pose", "arguments": {"query": "camera motion", "image_indices": [1, 2]}}
</tool_call>
<tool_response>
{"motion": {"yaw_deg": −24.2, "pitch_deg": 2.6, "roll_deg": 0.5, "t_m": [1.14, 0.04, −0.12]}}
</tool_response>
[Tool−call budget: at most 9 more tool calls may be made.]
<think>
t_m is the position of camera 2 in the frame of camera 1, with x to the right, y down and z forward. x = 1.14, so camera 2 is to the right. z = −0.12, so camera 2 is slightly behind camera 1. The camera also turned left by about 24 degrees, which fits with Image 2 looking at the chairs from the side. Right and behind: option A.
</think>
{"answer": "A"}

CAPAgent-8B, swapped tools (correct)

<think>
I need the position of camera 2 relative to camera 1 along x (left/right) and z (front/rear). I will query the camera motion and check both signs.
</think>
<tool_call>
{"name": "query_pose", "arguments": {"query": "camera motion", "image_indices": [1, 2], "check": [{"type": "camera_position", "axis": "x", "sign": "right"}, {"type": "camera_position", "axis": "z", "sign": "front"}]}}
</tool_call>
<tool_response>
{"motion": {"yaw_deg": −24.2, "pitch_deg": 2.6, "roll_deg": 0.5, "t_m": [1.14, 0.04, −0.12], "inlier_frac": 0.41, "reproj_px": 3.6, "overlap": 0.38}}
</tool_response>
[check position x right: margins (+7.9, +14.9) −> USE]
[check position z front: margins (−5.4, +3.0) −> RE−QUERY]
[Tool−call budget: at most 9 more tool calls may be made.]
<think>
Camera 2 is to the right; that part is decided. Front or rear is not: z = −0.12 m is small next to the sideways 1.14 m, and only 41% of the matches fit the pose. The two photos share little area. I will ask again with the images in the other order.
</think>
<tool_call>
{"name": "query_pose", "arguments": {"query": "camera motion", "image_indices": [2, 1], "check": {"type": "camera_position", "axis": "z", "sign": "behind"}}}
</tool_call>
<tool_response>
{"motion": {"yaw_deg": 23.8, "pitch_deg": −2.4, "roll_deg": −0.4, "t_m": [−1.19, −0.03, −0.29], "inlier_frac": 0.66, "reproj_px": 2.1, "overlap": 0.38}}
</tool_response>
[check position z behind: margins (+1.0, +4.8) −> USE]
[Tool−call budget: at most 8 more tool calls may be made.]
<think>
In the frame of camera 2, camera 1 is 1.19 m to the left and 0.29 m behind, and this is decided. Turning it back by the 24 degree yaw puts camera 2 at about x = +0.97 m and z = +0.75 m from camera 1: to the right and in front. Option D.
</think>
{"answer": "D"}

Analysis. MASt3R's first estimate got the sideways motion right and the forward motion wrong, which is typical when two views share little area: Image 1 is a close view of two chairs, and Image

2 shows the whole group from its side. SpatialCLI-8B followed the sign of a 12 cm value that sat next to a sideways offset ten times larger. CAPAGENT-8B's two checks separated the decided part of the reading (camera 2 is to the right) from the undecided part, and only the undecided part was queried again. The `overlap` field hardly changes when the image order is swapped (0.38 in both returns); the change in `inlier_frac` from 0.41 to 0.66 is what moved the second verdict to `USE`.

L.4 CASE 4 (FAILURE): UNDER-TRUSTING A USABLE POSE READING

Box L.4: Task (MMSI-Bench, Motion-Cam, item 213, swapped tools)



Image 1 Image 2

Two consecutive frames of a supermarket shelf stocked with cereal boxes and price labels. In Image 2 the boxes sit slightly further right in the frame. Many packages repeat along the shelf. This is panel (h) of Figure 9.

The images are taken continuously from a first-person perspective. In which direction is your viewpoint rotating?
Options: A: Down, B: Left, C: Right, D: Up

Ground truth. Correct option: **B**. Tools: camera motion is backed by MASt3R instead of VGGT.

SpatialCLI-8B, swapped tools (correct)

<think>
I need the rotation of the camera between the two images. Let me call query_pose with camera motion.
</think>
<tool_call>
{"name": "query_pose", "arguments": {"query": "camera motion", "image_indices": [1, 2]}}
</tool_call>
<tool_response>
{"motion": {"yaw_deg": −2.3, "pitch_deg": 0.4, "roll_deg": −0.1, "t_m": [0.03, 0.01, 0.02]}}
</tool_response>
[Tool−call budget: at most 9 more tool calls may be made.]
<think>
yaw_deg = −2.3, and negative yaw means the camera turned left. Pitch is much smaller, so the view is not mainly turning up or down. The boxes sitting further right in Image 2 also fit a turn to the left. Option B.
</think>
{"answer": "B"}

CAPAGENT-8B, swapped tools (incorrect)

<think>
I will query the camera motion and check whether the camera turned left.
</think>
<tool_call>
{"name": "query_pose", "arguments": {"query": "camera motion", "image_indices": [1, 2], "check": {"type": "camera_motion", "axis": "yaw", "sign": "left"}}}
</tool_call>
<tool_response>
{"motion": {"yaw_deg": −2.3, "pitch_deg": 0.4, "roll_deg": −0.1, "t_m": [0.03, 0.01, 0.02], "inlier_frac": 0.29, "reproj_px": 5.1, "overlap": 0.91}}
</tool_response>
[check motion yaw left: margins (−0.19, +0.65) −> RE−QUERY]
[Tool−call budget: at most 9 more tool calls may be made.]
<think>
The yaw is −2.3 degrees, but the check is not decided: only 29% of the matches fit the pose. I will try the other image order, then crops.
</think>
[calls 2 to 9: the same query with the images in the other order and on crops of the top shelf, the price labels, and the lower half; inlier_frac between 0.24 and 0.35, a turn to the left of 1.6 to 3.1 degrees, verdict RE-QUERY each time]
<tool_call>
{"name": "query_pose", "arguments": {"query": "camera motion", "image_indices": [1, 2], "check": {"type": "camera_motion", "axis": "yaw", "sign": "left"}}}
</tool_call>
<tool_response>
{"motion": {"yaw_deg": −2.0, "pitch_deg": 0.3, "roll_deg": −0.1, "t_m": [0.02, 0.01, 0.02], "inlier_frac": 0.30, "reproj_px": 4.9, "overlap": 0.91}}
</tool_response>
[check motion yaw left: margins (−0.24, +0.64) −> UNUSABLE]
[Tool−call budget exhausted: no more tool calls are allowed. Answer the question immediately using the available information.]
<think>
I could not get a usable pose reading, so I answer from the images. In Image 2 the boxes are further to the right. If the scene moves to the right, the view turned to the right. Option C.
</think>
{"answer": "C"}

Analysis. All ten readings agree that the camera turned left, by 1.6 to 3.1 degrees, and SpatialCLI-8B answered correctly from the first one. CAPAGENT-8B did not accept any of them: its predicted rotation radius stayed between 3.9 and 4.6 degrees, larger than the turn itself, because of the low `inlier_frac`. It spent the whole budget and then misread the image motion on its own. The two frames see almost the same shelf (`overlap` 0.91); the inlier fraction is low because identical boxes produce matches to the wrong copy, not because the pose is hard. The predicted radius comes from DUSt3R, the pose treatment in training, whose rotation errors grow quickly when the inlier fraction falls below about 0.4. MASt3R keeps the sign of small rotations right more often at the same inlier fraction, so here CAPAGENT-8B under-trusts a reading that was usable. This is the failure behind the Motion-Cam column, where CAPAGENT-8B trails GAR-8B with swapped tools (37.8 against 38.1, Table 1) and SpatialCLI-8B with the original tools (37.8 against 39.2, Table 14). CAPAGENT-8B discards the pose result on 14.2% of camera-motion questions, against 6.8% for GAR-8B. Among the Motion-Cam questions that CAPAGENT-8B gets wrong and SpatialCLI-8B gets right, 71% end with an `UNUSABLE` pose verdict after the budget runs out. It is the reverse of the too-narrow ranges on Metric3D v2 Small (Table 24): in both cases the agent applies the relation between cues and error that it saw on its training providers.

M LIMITATIONS AND BROADER IMPACT

Scope of the tasks. All decisions we study compare a continuous measurement with a threshold, which is what makes margins, and their changes, well defined. Many spatial questions have this form, as the gains on SpatialCLI-Bench and camera position show, but tools whose outputs are

masks, trajectories, or free-form descriptions do not reduce to one threshold. BOPASK, whose answers are point sets, moves by less than one point under any policy in Table 1.

What training needs. Same-scene pairs need at least two providers per capability run on the same scenes, and the reference margins need ground truth on a calibration split for each training provider. The calibration split is small (100 capture units), but it must come from the same kind of scenes as deployment, and our indoor, outdoor, and grounding policies are trained separately for this reason.

Range of reliability. Transfer is weakest where a swap changes reliability little or in a direction training never showed. Depth Anything V2 Large, with a radius 1.15 times the reference's, reaches 62.2% switch accuracy, and the three providers more reliable than the reference reach 59.9% (Table 21). Large changes transfer best: Metric3D v2 Small, 3.41 times the reference and noisier than any training treatment, reaches 78.8%, but most of its errors predict a range narrower than its calibrated one, with a radius near the widest seen in training (Table 24). Training with a wider spread of providers, including degraded versions of strong models and providers stronger than the reference, is the direct way to widen this range.

Errors the cues do not show. The agent judges a new model by how its cues relate to error, as the training models' cues did. A model whose error leaves no trace in the packet, such as a depth map that is uniformly 6% too large, looks like a reliable model to every policy that reads only the packet (Table 23). A few labelled scenes of the new model catch this case; a second, independent measurement of the same quantity would be the label-free way to expose it.

Designed cues. The packets use a few cues per capability. Table 4a shows that pairing needs informative cues, and in particular an image cue that is the same for every provider. Whether an agent that reads raw depth crops or confidence maps learns such cues itself is untested.

Broader impact. The work targets a practical safety problem: a robot or assistant that silently trusts a new perception model as much as the old one can act on a wrong measurement. Changing decisions when the evidence weakens, and asking for another measurement, reduces such commitments in our experiments. The same agent re-measures more often, which costs time, and a matched budget removes part of the benefit. The method does not certify a new model, and a deployment that changes a safety-critical perception component should validate it on its own data.