

AeroVerifier: A Typed Spatial Verifier for Frozen Vision-Language Models in Aerial Visual Question Answering

Jinfeng Chen Mengfan Yang Jianyu Zhu Zhicheng Zhou Jing Liang

University of Electronic Science and Technology of China

Chengdu, China

{chenjinfeng, 202311012240, 202511012320, zhicheng}@std.uestc.edu.cn

liangjing@uestc.edu.cn

Abstract

Drone-based applications such as navigation, inspection, and search depend on vision-language models (VLMs) that can answer spatial visual questions. The VLM faces three challenging tasks: grounding each referenced phrase to a scene object, choosing a viewpoint for spatial relations, and verifying the relation against accurate 3D geometry. In current frozen VLMs, all three tasks happen inside a single forward pass, and only the final answer is visible from outside. As a result, when that answer is wrong, we cannot tell which task failed. To address this, we introduce **AeroVerifier**, which exposes each of these three tasks as an explicit and checkable interface. Given a question, **AeroVerifier** compiles it into an executable typed program, runs the program over object-level 3D evidence, and returns the program answer together with stage-wise diagnostic signals to the base VLM as single-step revision feedback. On a 1000-row Open3D-VQA subset, both **AeroVerifier**'s program answer and its revision feedback yield higher answer accuracy than the Qwen2.5-VL-7B and GPT-5.4 API baselines.

1 Introduction

Drone-based applications such as navigation, inspection, and search require vision-language models (VLMs) that can answer visual question answering (VQA) queries about aerial spatial scenes (Zhang et al., 2025; Gao et al., 2024; Zhao et al., 2025; Zha et al., 2026). These questions test geometric relations between objects, rather than the location or orientation of a single object, including distance, camera-relative direction, and front/behind/left/right relations in 3D space. Correct spatial answers are important for target localization, collision-free planning, and structured scene description.

Although these questions appear simple, they are difficult for a frozen VLM. The model must solve

three coupled tasks: ground each phrase to a scene object, choose a viewpoint for the spatial relation, and verify the relation using 3D geometry. Take “*Is the white car to the left of the lamppost?*” as an example: the model must first bind “*the white car*” and “*the lamppost*” to the correct objects in the aerial scene, where nearby cars, poles, or small objects can create ambiguity. It must then decide what “*left*” means, because the answer can depend on whether the relation is interpreted from the image plane, the drone viewpoint, or another spatial frame. Finally, it must compare the two objects’ 3D positions under that frame, since a visually plausible 2D relation may not match the actual geometric relation in 3D space. Any error in these choices can easily flip the final answer.

However, in current frozen VLMs, all three tasks occur opaquely within a single forward pass. Consequently, only the final answer is visible, masking the root cause of the failure. Thus, when the answer is wrong, we cannot tell whether the failure came from grounding, viewpoint choice, or geometry. This opacity is especially problematic for aerial VQA, where a wrong spatial convention can produce a fluent answer that appears plausible but is geometrically false.

Existing work addresses this limited interpretability in two main ways. While spatially specialized VLMs and 3D-aware language models (Chen et al., 2024; Hong et al., 2023) improve accuracy via domain-specific finetuning, their language-to-geometry mapping remains a black box. Open-ended visual-programming and tool-use methods (Gupta and Kembhavi, 2023; Surís et al., 2023; Liang et al., 2023) expose intermediate computation, but their sampled free-form programs make coordinate conventions hard to control.

We take a third route. Instead of training a stronger model or sampling free-form programs, **AeroVerifier** (Fig. 1) compiles each question into a frame-committed typed program, executes it over

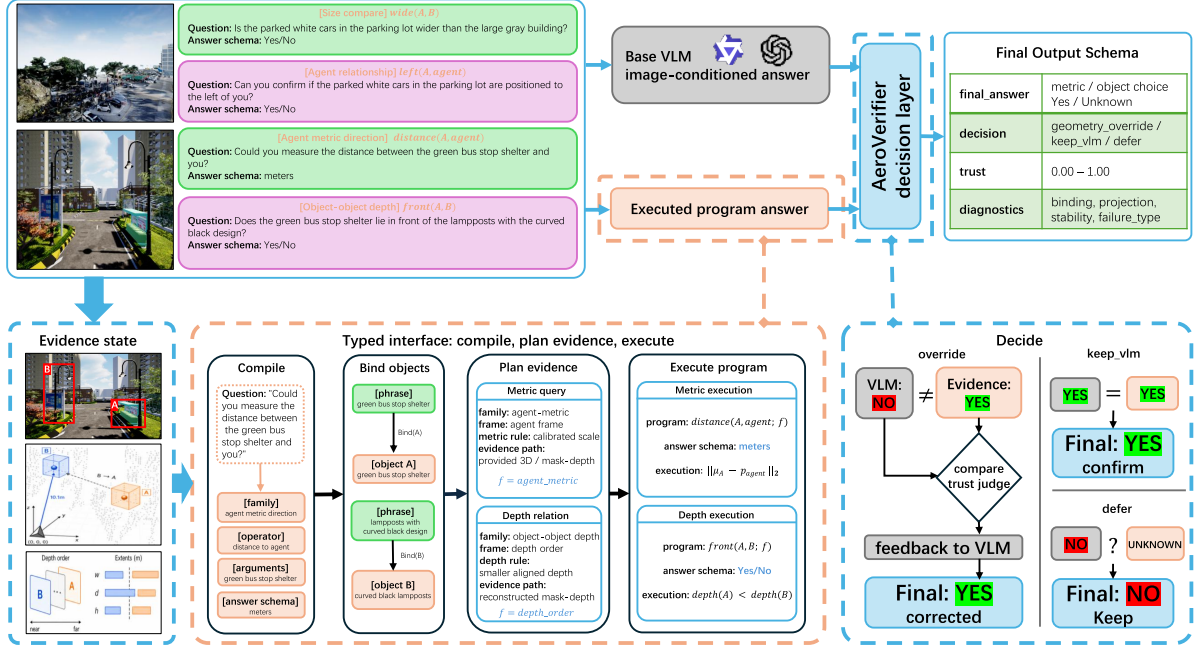


Figure 1: **AeroVerifier overview.** Four representative aerial questions (top-left, color-coded by family) drive the four-stage pipeline *compile* → *plan evidence* → *execute* → *decide*. The compiler commits a typed operator and coordinate frame, then dispatches to provided 3D (benchmark captions and centers/extents) or reconstructed mask-depth (CLIPSeg + SAM + aligned depth). The deterministic spatial executor returns an answer, trust diagnostics, and a typed failure type, fed back to the base VLM for single-step revision. The right-hand cards illustrate three outcome types: geometry override, keep VLM (under agreement or low-trust disagreement), and defer.

a unified object-level 3D evidence interface, and returns the program answer with diagnostics as single-step revision feedback. This turns spatial answering from an implicit prediction problem into an explicit verification problem.

Contributions.

- **A frame-committed typed verifier is introduced for aerial spatial questions.** To resolve semantic ambiguities, natural language queries are parsed into compact typed programs, which explicitly define the spatial operator, object arguments, and coordinate frame. By committing to a frame before execution, spatial relations are evaluated under fixed geometric rules. This structural design ensures reproducibility and explicitly isolates errors to either semantic parsing, frame selection, or 3D geometry.
- **A unified object-level 3D evidence interface is introduced.** Grounded objects are universally represented as a unified tuple, comprising the caption, 3D center, extent, bounding box, and evidence quality. By mapping all raw sensory inputs into this shared structure, the spatial executor is designed to operate independently of the un-

derlying evidence source. Consequently, spatial execution is fully decoupled from evidence construction, allowing real-world deployment bottlenecks to be directly measured.

- **An executable diagnostic feedback mechanism is developed to guide frozen VLM revision.** Instead of merely outputting a program answer, the verifier generates rich diagnostic signals, including binding quality, margin strength, and structural consistency. These explicit reasoning traces function as an evidence-grounded prompt injection. Consequently, frozen VLMs are provided with concrete geometric facts for answer revision, successfully overcoming the limitations of opaque, blind self-correction.

2 Related Work

Spatial VLMs and aerial benchmarks. Spatial reasoning has been studied in both general VQA benchmarks and 3D scene understanding. VQA, CLEVR, GQA, and ScanQA (Antol et al., 2015; Johnson et al., 2017; Hudson and Manning, 2019; Azuma et al., 2022) separate recognition from compositional and geometric reasoning. Recent aerial and embodied benchmarks, including Open3D-

VQA, EmbodiedCity, UrbanVideo-Bench, and Air-CopBench (Zhang et al., 2025; Gao et al., 2024; Zhao et al., 2025; Zha et al., 2026), make viewpoint conventions, depth ordering, and metric distance central to the answer. Spatially specialized VLMs and 3D-aware language models (Chen et al., 2024; Hong et al., 2023) address this challenge with extra supervision, 3D inputs, or model adaptation. Our focus is different. Instead of training a stronger aerial model, we expose the intermediate commitments that a frozen VLM normally leaves implicit: object binding, coordinate frame, evidence source, and final answer routing.

Program-aided reasoning and tool use.

Program-aided methods translate model outputs into code, module calls, or tool-use traces. Visual Programming, ViperGPT, Toolformer, ReAct, Chameleon, and Code-as-Policies (Gupta and Kembhavi, 2023; Surís et al., 2023; Schick et al., 2023; Yao et al., 2023; Lu et al., 2023; Liang et al., 2023) are flexible because the model can choose intermediate computations. For aerial spatial VQA, however, this flexibility creates risk. Sampled programs may change across runs. Helper functions may encode unclear coordinate conventions. Object grounding can also be mixed with answer generation. AeroVerifier instead uses a small typed operator language whose frame semantics are fixed before execution. This limits open-ended program synthesis, but it makes every executable answer reproducible. It also makes parse and evidence failures attributable to a specific stage.

Selective prediction and confidence signals.

Our answer-selection layer is related to selective prediction, calibrated self-knowledge, and process-level supervision (Geifman and El-Yaniv, 2017; Kadavath et al., 2022; Lightman et al., 2024). These areas study when a model should answer, revise, or abstain. They often summarize reliability with scalar confidence, self-assessment, or a learned reward. In contrast, our diagnostics are tied to the spatial computation itself. They measure binding quality, object evidence quality, margin strength, reciprocal consistency, perturbation stability, and operator support. The goal is therefore twofold: decide whether to trust an answer, and provide a frozen VLM with concrete evidence for when a program answer should replace its original response.

3 AeroVerifier

3.1 Overview and Task Formulation

AeroVerifier is built around the three contributions above: a frame-committed typed verifier, a unified object-level 3D evidence interface, and executable diagnostic feedback for frozen VLMs. It makes the language-to-evidence interface explicit through four stages (Fig. 1): question compilation into a typed program, frame-aware evidence planning, deterministic execution, and diagnostic-guided answer selection. Given an aerial image, a question, the base VLM answer, and detected scene objects with 3D positions and sizes, AeroVerifier checks whether the question can be converted into an executable spatial program. If execution succeeds, it decides whether the program answer should replace the VLM answer. Formally, given an aerial image I , a question q , a base VLM answer y_v , and an evidence state

$$\mathcal{E} = \{(c_i, \boldsymbol{\mu}_i, \mathbf{s}_i, b_i, \rho_i)\}_{i=1}^N, \quad (1)$$

AeroVerifier decides whether q compiles into an executable typed program. If it does, AeroVerifier decides whether the program answer y_g should override y_v . Here N is the number of evidence objects. Each c_i is an object caption, $\boldsymbol{\mu}_i \in \mathbb{R}^3$ is a robust 3D center, $\mathbf{s}_i \in \mathbb{R}^3$ is an object extent, b_i is an optional 3D bounding box, and $\rho_i \in [0, 1]$ is an evidence-quality score. The same tuple shape is preserved across evidence regimes; only the source of these fields changes.

This formulation operationalizes the three tasks in the introduction. Object binding matches a to elements of $\mathcal{E}$. Spatial-frame commitment is recorded in f . Finally, 3D-evidence verification uses the fields in $\mathcal{E}$. Because these choices are represented before execution, later errors can be traced to a concrete stage instead of to an opaque model answer. The key design is that these steps are not tied to one evidence source or one VLM backbone. The same typed program can execute over different object-level 3D regimes, and the resulting answer can be routed back to any frozen VLM as feedback.

The task families used by the compiler are summarized in Tab. 1; LRUD denotes left/right/up/down image-plane relations.

3.2 Question Template Compilation

The compiler maps a question to a typed program

$$\pi = (t, \omega, \mathbf{a}, f, C), \quad (2)$$

Family	Frame	Operator(s)	Answer	Overall	Exec.	Δ rand.
Agent metric	agent-centric	direction(A, agent)	clock	82.04	82.04	-5.39
Agent relation	agent-centric	left(A, agent)	yes / no	64.67	77.14	-48.50
Object LRUD	image-plane	left(A, B), right(A, B)	yes / no	81.93	90.07	0.00
Object depth	depth ordering	front(A, B), behind(A, B)	yes / no	55.09	89.32	0.00
Size compare	semantic extent	tall(A, B), wide(A, B)	object choice	70.48	70.48	0.00
Numeric metric	ground-metric	distance(A, B), height(A)	scalar	40.72	48.23	-6.59

Table 1: **Task-family overview on the balanced Open3D-VQA split.** Each family contains 166–167 rows. *Overall Acc.* is verifier accuracy over all rows in the family, so parse and evidence failures count as wrong. *Exec. Acc.* is conditional accuracy over rows that both parse and produce executable evidence. Δ rand. is the change in *Overall Acc.* when only the calibrated agent frame is replaced by a deterministic random egocentric/allocentric flip; 0 indicates families whose operators do not depend on that frame choice.

where π is the compiled program, t is the relation family, ω is the operator, $\mathbf{a} = [a_1, \dots, a_K]$ is the list of K natural-language object arguments, f is the committed coordinate frame, and $\mathcal{C}$ is an optional choice set for multiple-choice questions.

The compiler first routes a question to a family using template metadata and surface cues. It then extracts typed arguments and *commits a coordinate frame* (agent-egocentric, object-relative, or depth-ordering) according to the relation expressed. Once f is fixed, the downstream geometric interpretation is deterministic. The headline configuration uses the template parser. A constrained GPT-5.4 API JSON parser gives higher validity but lower end-to-end verifier accuracy on this benchmark.

3.3 Frame-Aware 3D Evidence Construction

This stage implements the unified object-level 3D evidence interface. It produces the evidence state $\mathcal{E}$ of Eq. 1 and aligns the evidence to the frame fixed in §3.2. The important point is that all evidence regimes fill the same tuple shape. We first describe how each object tuple is *constructed* from raw inputs. We then describe how the spatial signal is *selected* for the committed frame.

Constructing the object tuple. Both evidence regimes use the same tuple $(c_i, \mu_i, s_i, b_i, \rho_i)$. They differ only in where the 3D points come from. In the *provided* regime, points are read from the benchmark’s per-object segmented point clouds. In the deployment-realistic *reconstructed* regime, no benchmark 3D is used. Instead, each program argument becomes a text phrase. CLIPSeg (Lüdecke and Ecker, 2022) grounds the phrase to a soft region (threshold 0.45), and the Segment Anything Model (SAM) vit_h (Kirillov et al., 2023) refines the region into a mask. The masked pixels

of the released aligned depth map are then back-projected into a per-object point cloud through a pinhole camera with horizontal field of view 90° ,

$$\mathbf{p}(u, v) = d(u, v) K^{-1}[u, v, 1]^\top, \quad (3)$$

where $\mathbf{p}(u, v) \in \mathbb{R}^3$ is the back-projected 3D point for pixel (u, v) , $d(u, v)$ is the aligned depth at that pixel, and $[u, v, 1]^\top$ is the homogeneous pixel coordinate transformed by the intrinsic matrix K . Because Open3D-VQA does not release per-image intrinsics, we assume a fixed-FOV pinhole camera and conservatively gate metric queries (§3.5). From the resulting points, the robust center μ_i is the per-axis median. The extent s_i and box b_i are read from the 2nd–98th percentile bounds. The quality score $\rho_i = 1 - \exp(-N_i/500)$ grows with the number of valid points N_i and is down-weighted by grounding confidence in the reconstructed regime. Thus, the tuple shape is preserved across regimes; only its source varies (Fig. 2).

Selecting the frame-aligned signal. Given the tuple, the executor must still read it under the committed frame. We call this frame-aligned selection *viewer-frame planning*. For provided 3D, it selects calibrated centers, extents, and per-operator projection checks. For reconstructed mask-depth, it uses image-plane centroids for LRUD and agent-relative relations. It also treats smaller aligned depth as closer for front/behind, and routes metric queries through the guarded pathway when scale is unreliable. In short, compilation decides what the question means, while this step decides how the available evidence should instantiate that meaning.

Each natural-language argument a_j is bound to the evidence object with the highest lexical-overlap score ϕ_{lex} . This score combines phrase coverage with stop-word-stripped Jaccard at weight

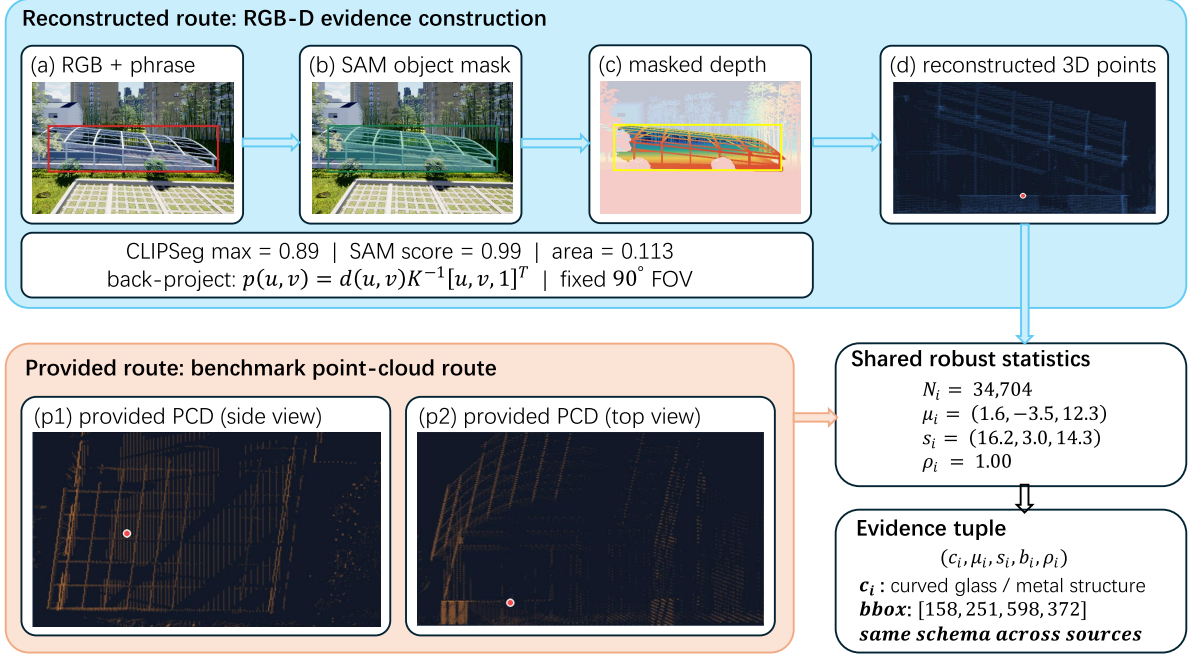


Figure 2: **Constructing one object’s evidence tuple.** In the reconstructed regime (top), a program-argument phrase is grounded by CLIPSeg, refined by SAM, and back-projected from the aligned depth map via Eq. 3 (pinhole camera, 90° FOV, no per-image intrinsics). In the provided regime (bottom), the benchmark point cloud enters at the same node. Both paths produce the same tuple fields: caption, robust center, extent, optional 3D box, and evidence quality.

$\alpha = 0.55$ (formal definition in App. C). We freeze α on a disjoint 200-row development split, which also fixes the override thresholds in Tab. 9. Binding error changes by less than 1 pp across $\alpha \in \{0.40, \dots, 0.60\}$.

3.4 Typed Program Execution

Execution proceeds in three deterministic steps. First, the bound evidence object o_j^* is retrieved for each argument a_j . Second, the program is dispatched according to the committed frame f , which may be viewer-relative, agent-egocentric, object-relational, ground-metric, or semantic-extent. Third, the operator is executed over the resulting frame-aligned evidence state.

For binary qualitative relations, we abstract execution as a signed margin between two bound objects o_a and o_b :

$$m_\omega(o_a, o_b; f) = \mathbf{w}_\omega(f)^\top (\boldsymbol{\mu}_a - \boldsymbol{\mu}_b), \quad (4)$$

where o_a and o_b are the two bound objects, and $\boldsymbol{\mu}_a$ and $\boldsymbol{\mu}_b$ are their centers. The vector $\mathbf{w}_\omega(f) \in \{-1, 0, +1\}^3$ selects the signed axis implied by operator ω under frame f . The executed verdict is

$$\hat{z}_\omega = \mathbb{I}[m_\omega > 0], \quad (5)$$

where $\mathbb{I}[\cdot]$ is the indicator function. This verdict is used directly for yes/no relations and is mapped to the appropriate textual answer for object-choice or labeled options. To avoid over-trusting near ties, we also compute a bounded strength score

$$q_{\text{margin}} = 1 - \exp(-|m_\omega|/2). \quad (6)$$

Its threshold is fixed on a disjoint development split. Size operators use semantic extents derived from s rather than centers $\boldsymbol{\mu}$. Metric operators return scalar distances or extents. Projection agreement, reciprocal consistency, and counterfactual stability are evaluated as additional diagnostics (App. C).

3.5 Diagnostics and Answer Selection

The final stage turns deterministic execution into diagnostic feedback for the frozen VLM. We treat per-image evidence as a spatial graph whose nodes are evidence objects and whose edges are deterministic spatial relations. A typed program is a query over this graph that returns three outputs: a candidate program answer y_g , a typed failure mode, and a six-dimensional diagnostic vector

$$\boldsymbol{\kappa}(\pi, \mathcal{E}) = (q_{\text{bind}}, q_{\text{obj}}, q_{\text{mar}}, q_{\text{rec}}, q_{\text{cf}}, q_{\text{op}}) \quad (7)$$

measuring argument-binding quality, per-object evidence quality, signed-margin strength, reciprocal

consistency, counterfactual stability, and operator support (App. C). We average these into a graph-evidence score

$$C(\pi, \mathcal{E}) = \frac{1}{6} \sum_k \kappa_k \in [0, 1]. \quad (8)$$

Projection agreement and family-specific hard rules are checked separately, and the final fusion policy combines C with the operator-specific trust score.

The rule-fusion policy emits y_g only when execution succeeds, graph evidence and operator trust are sufficient, and the family-specific rule clears its threshold. Otherwise, AeroVerifier keeps the base answer y_v or defers in conservatively gated numeric cases. All thresholds are frozen on a held-in-split disjoint from evaluation. The pipeline is deterministic given $\mathcal{E}$ and q : no language-model sampling occurs after compilation.

4 Experiments

4.1 Setup

The experiments test three core design choices: the frame-committed typed verifier, the unified object-level 3D evidence interface, and executable diagnostic feedback for frozen-VLM revision. We first measure the verifier under controlled evidence, then test VLM revision, and finally analyze which evidence source limits deployment.

Datasets and evidence regimes. We evaluate on Open3D-VQA using a 1000-row balanced subset. The subset is stratified across the six families introduced in §3.2. For cross-benchmark transfer, we use 807 AirCopBench Sim3/5/6 multi-UAV multiple-choice rows with benchmark-native semantic fields. The primary controlled setting is *provided evidence*. It includes the red-green-blue (RGB) image, question, base VLM answer, and Open3D-VQA object captions, centers, extents, and view metadata. We also evaluate reconstructed mask-depth evidence on the same rows. *Mask-depth 3D* reconstructs object-level 3D from phrase-grounded masks and the aligned depth map (§3.3), without using benchmark point clouds. AeroVerifier requires no task-specific training and is applied directly to the base VLM answer.

Compared methods. On the same 1000 matched rows, we compare AeroVerifier with four groups of methods: image-only Qwen2.5-VL-7B and GPT-5.4 API, prompt baselines with serialized 3D and

chain-of-thought, generated-program baselines (Visual Programming, Code-as-Policies-style, and ViperGPT-style Python), and AeroVerifier’s deterministic answer and single-step revision. Revision uses a matched VLM-call budget; frame ablations are in Tab. 4, with supporting diagnostics in App. F. All VLMs use deterministic decoding.

Metrics and compute. Beyond accuracy, we report fix/harm counts, McNemar tests with Cohen’s h , per-family accuracy, the Overall-vs-Executable gap, and typed failure modes. Every revision condition uses exactly two base-VLM forward passes per row. The executor is deterministic CPU Python with negligible per-row cost; full settings and matcher validation are in App. E–E.1.

4.2 Evidence Interface Ceiling

With benchmark captions and object-level 3D, Table 2 holds evidence fixed while comparing spatial commitment and execution. Under provided 3D, AeroVerifier reaches 68.70%, above GPT-5.4 API prompt (51.10–51.90%) and generated-program (40.20–54.30%) baselines, and Qwen2.5-VL-7B prompt (32.30–34.70%) and generated-program (11.80–35.70%) baselines. Under mask-depth it reaches 45.70%, close to the strongest GPT-5.4 API open-program baseline (45.50%). These results support the typed-verifier design: when frame semantics and execution are fixed, it is more reliable than prompting or sampled programs over the same evidence.

With verifier feedback, GPT-5.4 API reaches 45.60/40.40% and Qwen2.5-VL-7B 36.00/28.60% under provided/mask-depth 3D; under mask-depth, weaker generated-program interfaces leave up to 80.3% Unknown or empty outputs.

Scale check. The same six-family protocol is also evaluated on a larger balanced split with benchmark-provided 3D evidence and the same verifier-centric answer-selection policy. Table 3 reports the result separately from the matched comparison above.

Frame commitment is important. Adding axis calibration and agent-frame dispatch raises deterministic program-only accuracy from 56.20% to 68.70% (+12.50 pp; Tab. 4). Replacing only the calibrated agent frame with a deterministic random egocentric/allocentric flip lowers accuracy by 10.1 pp. The per-family Δ rand. column of Tab. 1 shows that this loss is concentrated in agent-frame-

Method	GPT-5.4 API Acc./Unk. (%)		Qwen2.5-VL-7B Acc./Unk. (%)	
	Provided 3D	Mask-depth	Provided 3D	Mask-depth
<i>Prompt-only baselines</i>				
Image-only	35.20 / 1.5		21.80 / 0.0	
Serialized 3D prompt	51.90 / 7.0	42.30 / 19.4	34.70 / 0.0	34.00 / 0.8
Chain-of-thought (Wei et al., 2022)	51.10 / 8.0	42.80 / 20.3	32.30 / 0.0	29.70 / 0.4
<i>Generated-program baselines</i>				
Visual Programming (Gupta and Kembhavi, 2023)	40.20 / 11.9	35.20 / 25.6	11.80 / 77.1	8.70 / 80.3
Code-as-Policies-style (Liang et al., 2023)	42.70 / 11.6	36.10 / 23.1	35.70 / 16.4	30.30 / 30.0
ViperGPT-style Python (Surís et al., 2023)	54.30 / 2.2	45.50 / 13.2	35.30 / 15.5	29.50 / 25.7
<i>Typed verifier</i>				
AeroVerifier program answer	68.70 / 3.4	45.70 / 12.7	68.70 / 3.4	45.70 / 12.7
Program-guided revision	45.60 / 1.3	40.40 / 1.1	36.00 / 0.0	28.60 / 30.6
Routing oracle	78.70 / n/a	59.20 / n/a	77.20 / n/a	56.10 / n/a

Table 2: **Backbone-matched comparison on the balanced 1000-row Open3D-VQA subset.** Cells report *Acc./Unk.* (answer accuracy / fraction of Unknown or empty outputs) under benchmark-provided 3D and reconstructed mask-depth evidence (§3.3). Prompt-only baselines answer from the image and/or serialized evidence; generated-program baselines generate and execute a program over the same object-level evidence. *AeroVerifier program answer* is the fixed deterministic verifier-centric answer-selection output; Tab. 1 reports raw per-family verifier accuracy before this routing. It uses no new LM call and is therefore identical across backbones; the revision row is a separate two-pass adoption test of whether each frozen VLM uses verifier feedback, with mask-depth paired *fix/harm* in Tab. 5. Routing-oracle rows are counterfactual upper bounds rather than deployed outputs, so their Unk. entry is not applicable and is shown as *n/a*.

Setting	Rows	Acc. (%)	Fix vs. base (%)	Harm vs. base (%)
Qwen image-only	30,000	20.57	–	–
AeroVerifier (provided 3D)	30,000	65.87	46.97	1.67

Table 3: **Auxiliary scale check.** Balanced Open3D-VQA results with 5,000 rows per spatial family, using benchmark-provided 3D evidence and the same verifier-centric answer-selection policy as the main provided-3D result. Fix and harm are percentages relative to the Qwen2.5-VL-7B image-only base.

sensitive families, rather than object–object relations. Therefore, the gain comes from the first contribution: committing the correct spatial convention before execution, not only from exposing 3D coordinates.

4.3 Typed Programs as VLM Feedback

We next evaluate the deployment-realistic case. This experiment connects the second and third contributions: the verifier must work with reconstructed object-level 3D evidence, and the frozen VLM must decide whether to adopt the verifier’s diagnostic feedback. Here, the verifier uses reconstructed mask-depth evidence and returns its output to the base VLM as single-step revision feedback. The no-program control receives only the image, question, and original VLM answer. The program-guided variant additionally receives the executed

Variant	Axis	Agent	Scale	Program Acc.	Inc. Δ
No frame calibration	–	–	–	56.20	–
+ axis	✓	–	–	61.10	+4.90
+ axis + agent (ours)	✓	✓	–	68.70	+7.60
+ axis + agent + scale	✓	✓	✓	67.60	–1.10

Table 4: **Frame-commitment ablation on the same 1000 provided-evidence rows.** *Program Acc.* is deterministic program accuracy with no VLM call; parser, provided 3D evidence, executor, and matcher are held fixed; only axis calibration, agent-frame dispatch, and optional metric-scale calibration vary. The per-family random-frame counterfactual is reported in the Δ rand. column of Tab. 1.

typed-program answer and its diagnostics (App. B), but never the gold label.

In the budget-matched revision setting, program-guided revision tests whether executed geometry and diagnostics can improve frozen VLMs. Under deployment-realistic mask-depth, it raises Qwen2.5-VL-7B from 21.80% to 28.60% and GPT-5.4 API from 35.20% to 40.40%. Image-only self-revision gives smaller and less consistent changes (+4.40 pp on Qwen2.5-VL-7B; –1.20 pp on GPT-5.4 API). With provided 3D, the same protocol reaches 36.00% on Qwen2.5-VL-7B and 45.60% on GPT-5.4 API. Table 5 shows that mask-depth program-guided revision produces more fixes than harms on both backbones.

Backbone	Revision	Δ pp	Fix	Harm
GPT-5.4 API	self-rev.	−1.20	48	60
	prog.-guided	+5.20	90	38
Qwen2.5-VL-7B	self-rev.	+4.40	123	79
	prog.-guided	+6.8	124	57

Table 5: **Paired feedback statistics on the same 1000 Open3D-VQA rows under reconstructed mask-depth evidence.** Δ pp is the paired accuracy gain over the base VLM. Fix/Harm count rows the base VLM gets wrong/right that the revision corrects/regresses. Bold marks program-guided revision; significance tests and effect sizes are reported in App. F.

Backbone	Δ pp	Fix	Harm	McNemar p
Qwen2.5-VL-7B	+1.30	62	49	0.25
GPT-5.4 API	−1.30	1	14	9.8×10^{-4}

Table 6: **Compile-without-execute control.** The typed program is retained but its executed answer and diagnostics are withheld.

The executor, not the typed-program text, accounts for the feedback gain. We test a *compile-but-don’t-execute* control. In this control, the typed program and reliability note remain, but the program answer and diagnostics are blanked. Qwen2.5-VL-7B shifts only +1.30 pp, with 62 fixes and 49 harms (McNemar $p = 0.25$). GPT-5.4 API shifts −1.30 pp, with 1 fix and 14 harms (McNemar $p = 9.8 \times 10^{-4}$). Both results are far below the full program-guided gains. Thus, the gains come from executed evidence rather than from program text as extra context.

Corrupted evidence. Table 7 reports provided-3D perturbations without extra VLM/API calls.

4.4 Cross-Benchmark Protocol Replication

On 807 AirCopBench Sim3/5/6 rows with benchmark-native semantic fields, deterministic execution reaches 61.59%, while program-guided revision raises Qwen2.5-VL-7B from 44.24% to 60.72% and GPT-5.4 API from 50.19% to 71.75%.

4.5 Error Analysis

Errors follow a clear pattern. In total, 75.1% of rows produce no failure. The remaining 24.9% are dominated by parse errors (11.7%) and projection inconsistencies (11.3%). Depth questions are limited by parser and projection availability rather than the executor, as shown by the Overall-vs-Executable gap of 34.2 pp in Tab. 1. The numeric family remains weakest at 48.23% on parse-

Corruption	GPT-5.4 base	Qwen2.5-VL-7B base
5% center perturbation	−3.30	−3.30
20% extent perturbation	−1.10	−1.10
20% label swaps	−2.40	−2.40
20% object dropout	−8.90	−10.40

Table 7: **Evidence-corruption audit.** Accuracy change (percentage points) from clean provided 3D.

successful rows, where the conservative policy often defers. Full breakdown in Tab. 12; qualitative examples in App. D.

This pattern separates two bottlenecks: parsing and frame agreement determine whether a program can execute, while evidence quality determines whether its result is trustworthy. The matched controls therefore support treating perception and verification as separate deployment stages.

Executable-row scores therefore isolate geometric reasoning, while all-row accuracy also reflects parser and evidence availability.

5 Conclusion

This study aims to enable the reliable deployment of aerial spatial VQA in drone-based navigation, inspection, and search. In such domains, spatial predictions must be not only geometrically accurate but also explicitly attributable to concrete visual evidence. The central challenge is that frozen VLMs combine object grounding, viewpoint selection, and 3D geometric verification into a single opaque answer, making failures difficult to diagnose or correct.

To address this problem, we introduce **AeroVerifier**, an explicit spatial verification framework for frozen VLMs. On Open3D-VQA, its program answer reaches 68.70% accuracy with benchmark-provided 3D and 45.70% with reconstructed mask-depth evidence, outperforming prompt-only and generated-program baselines. As single-step revision feedback, program-guided revision raises Qwen2.5-VL-7B from 21.80% to 28.60% and GPT-5.4 API from 35.20% to 40.40% under reconstructed mask-depth evidence. The same pattern is reproduced on AirCopBench. These results show that explicit, evidence-based verification can improve aerial VQA reliability without retraining the underlying VLM, while object-level RGB-to-3D evidence construction remains the main deployment bottleneck.

Limitations

Our evaluation focuses on Open3D-VQA and AirCopBench Sim3/5/6. These two benchmarks provide structured spatial evidence, which enables the controlled paired comparisons used in this paper. Extending the interface to broader free-form aerial or video questions would require new parsers and operators. In the reconstructed-evidence setting, Open3D-VQA does not release per-image intrinsics. Therefore, metric-scale answers are treated conservatively. Mask, depth, and grounding errors can also degrade the object tuples consumed by the verifier, and remain the main bottlenecks for RGB-only deployment. Finally, all headline runs use deterministic decoding on fixed evaluation splits. The reported numbers should therefore be read as controlled paired comparisons, not as estimates over many decoding seeds.

Ethics Statement

This work uses existing aerial and embodied VQA benchmarks, Open3D-VQA (Zhang et al., 2025) and AirCopBench (Zha et al., 2026), together with publicly released CLIPSeg (Lüddecke and Ecker, 2022) and SAM (Kirillov et al., 2023) checkpoints. We collect no new imagery, annotations, personally identifying information, or human-subject data. The verifier is intended as a reliability layer for benchmark analysis, not as an autonomous navigation or surveillance system; real-world aerial deployment would require separate safety, privacy, and domain calibration review.

Acknowledgments

AI assistants were used for language editing and figure mock-up brainstorming; all scientific claims and final manuscript contents were verified by the authors. This work was supported in part by the National Natural Science Foundation of China under Grant 62471118 and 111 Project under Grant B17008.

References

- Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. [VQA: Visual question answering](#). In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433.
- Daichi Azuma, Taiki Miyanishi, Shuhei Kurita, and Motoaki Kawanabe. 2022. [ScanQA: 3d question answering for spatial scene understanding](#). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19129–19139.
- Boyuan Chen, Zhuo Xu, Sean Kirmani, Brian Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. 2024. [Spatialvlm: Endowing vision-language models with spatial reasoning capabilities](#). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14455–14465.
- Chen Gao, Baining Zhao, Weichen Zhang, Jinzhu Mao, Jun Zhang, Zhiheng Zheng, Fanhang Man, Jianjie Fang, Zile Zhou, Jinqiang Cui, Xinlei Chen, and Yong Li. 2024. [Embodiedcity: A benchmark platform for embodied agent in real-world city environment](#). Preprint, arXiv:2410.09604.
- Yonatan Geifman and Ran El-Yaniv. 2017. [Selective classification for deep neural networks](#). In *Advances in Neural Information Processing Systems*, volume 30.
- Tanmay Gupta and Aniruddha Kembhavi. 2023. [Visual programming: Compositional visual reasoning without training](#). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14953–14962.
- Yining Hong, Haoyu Zhen, Peihao Chen, Shuhong Zheng, Yilun Du, Zhenfang Chen, and Chuang Gan. 2023. [3D-LLM: Injecting the 3d world into large language models](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 20482–20497.
- Drew A. Hudson and Christopher D. Manning. 2019. [GQA: A new dataset for real-world visual reasoning and compositional question answering](#). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6700–6709.
- Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. 2017. [CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning](#). In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910.
- Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, and 17 others. 2022. [Language models \(mostly\) know what they know](#). Preprint, arXiv:2207.05221.
- Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. 2023. [Segment anything](#). In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026.

- Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. 2023. [Code as policies: Language model programs for embodied control](#). In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 9493–9500.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. [Let’s verify step by step](#). In *International Conference on Learning Representations*.
- Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. [Chameleon: Plug-and-play compositional reasoning with large language models](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 5514–5530.
- Timo Lüddecke and Alexander Ecker. 2022. [Image segmentation using text and image prompts](#). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7086–7096.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. [Toolformer: Language models can teach themselves to use tools](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 68539–68551. Curran Associates, Inc.
- Dídac Surís, Sachit Menon, and Carl Vondrick. 2023. [ViperGPT: Visual inference via python execution for reasoning](#). In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11888–11898.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. [Chain-of-thought prompting elicits reasoning in large language models](#). In *Advances in Neural Information Processing Systems*.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. [React: Synergizing reasoning and acting in language models](#). In *The Eleventh International Conference on Learning Representations*.
- Jirong Zha, Yuxuan Fan, Tianyu Zhang, Geng Chen, Yingfeng Chen, Chen Gao, and Xinlei Chen. 2026. [Aircopbench: A benchmark for multi-drone collaborative embodied perception and reasoning](#). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40(2), pages 1507–1515.
- Weichen Zhang, Zile Zhou, Xin Zeng, Xuchen Liu, Jianjie Fang, Chen Gao, Yong Li, Jinqiang Cui, Xinlei Chen, and Xiao-Ping Zhang. 2025. [Open3d-vqa: A benchmark for comprehensive spatial reasoning with multimodal large language model in open space](#). Preprint, arXiv:2503.11094.
- Baining Zhao, Jianjie Fang, Zichao Dai, Ziyu Wang, Jirong Zha, Weichen Zhang, Chen Gao, Yue Wang, Jinqiang Cui, Xinlei Chen, and Yong Li. 2025. [Urbanvideo-bench: Benchmarking vision-language models on embodied intelligence with video data in urban spaces](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32400–32423, Vienna, Austria. Association for Computational Linguistics.

A Data Selection and Scoring

Balanced selection. The verifier evaluation uses a 1000-row balanced subset from the Open3D-VQA evaluation pool, stratified across the six spatial families of Tab. 1 with 167 rows each for Agent metric, Agent relation, Numeric metric, and Object depth, and 166 rows each for Object LRUD and Size compare. Selection uses fixed seed, contains no duplicate question texts, and preserves scene coverage.

Split	Rows	Coverage
Open3D-VQA balanced	1,000	Wuhan 336; Campus 334; Residence 330
AirCopBench Sim3/5/6	807	multi-UAV multiple choice

Table 8: **Evaluation splits.** The Open3D-VQA split is fixed, balanced across six spatial families, and contains no duplicate question text.

The six Open3D-VQA families are balanced to 166–167 rows each (Tab. 1); AirCopBench is evaluated separately because its benchmark and evidence schema differ.

Scene coverage. The balanced split spans EmbodiedCity/Wuhan (336 rows), UrbanScene/Campus (334), and UrbanScene/Residence (330). The AirCopBench replication in §4.4 is kept separate because it swaps both the benchmark and the structured-evidence source.

Answer matching. Because answers include yes/no predicates, object phrases, choice labels, and numeric strings, evaluation uses answer-type-specific normalization rather than raw string equality. The same approximate matcher is used for Qwen2.5-VL-7B, GPT-5.4 API baselines, and verifier outputs. It normalizes multiple-choice prefixes, handles yes/no polarity, and tolerates lexical overlap for open object names. Numeric answers are still treated conservatively because small metric deviations can fail strict benchmark-style scoring. This is why numeric guarded rows are reported separately.

B Prompt Templates

All VLM baselines use deterministic decoding where supported. We retain only the two prompt forms most needed to interpret the main results: the benchmark-native image-only prompt and the program-guided revision prompt. The remaining variants differ only in which structured fields are exposed to the model and are summarized after the quoted templates.

Original Open3D-VQA image-only prompt.

The matched GPT-5.4 API and Qwen2.5-VL-7B image-only baselines use the original benchmark instruction below, followed by the question.

You are an assistant who perfectly answer question in urban environment. Only based on the image, you should directly answer the height, width, volume and distance question with exact number Answer the distance without output intermediate process. You should answer the direction question in the direction of the clock with taking your front as 12 o'clock, your left as 9 o'clock, and your right as 3 o'clock.

Question: {question}

Program-guided revision prompt (Open3D-VQA).

The program-guided revision prompt used in our experiments is shown below. The user message contains the RGB image followed by this text. The answer schema note is filled according to the question type, and the reliability note is generated by the verifier as strong, moderate, or weak.

You are revising one aerial spatial VQA answer. Use the image and the structured feedback below. Return only the final answer. Do not explain.

Output rules:

1. {answer_schema_note}
2. Do not output A.xxx, B.xxx, C.xxx, or D.Unknown unless the original question explicitly lists A/B/C/D options.
3. If Program answer is UNKNOWN, output the Original answer exactly.
4. If Reliability is weak, keep the Original answer unless the image and program clearly support a simple yes/no relation.
5. If Program answer is reliable and directly contradicts the Original answer, revise to the Program answer. Otherwise keep the Original answer.

Question: {question}

Original answer: {original}

Program family/operator: {family} / {operator}

Program answer: {program_answer}
{reliability_note}

Evidence summary:

- object 0: {caption}; bbox={bbox};
pseudo_center={center}; quality={quality}
... (up to 3 grounded objects)

Final answer:

Other prompt variants. The self-revision control reuses the same answer-schema constraints but omits the program answer, reliability note, and evidence summary. Thus, any gain must come from re-inspecting the image alone. The true structured-3D baseline serializes the same object-level geometry consumed by the executor and asks the VLM for a direct final answer. The AirCopBench prompts mirror the same three conditions: base, self-revision,

and program-guided revision. They use a text-only multiple-choice schema with one option per line. Full prompt strings remain in the released code.

C Spatial Operators

Object binding and execution. Argument phrases a_j are bound to evidence objects o_i by argmax lexical-overlap scoring,

$$\phi_{\text{lex}}(a_j, c_i) = \alpha \text{cov}(a_j, c_i) + (1 - \alpha) J(a_j, c_i), \quad (9)$$

with cov phrase coverage, J Jaccard over stop-word-removed token sets, and $\alpha = 0.55$; zero score triggers `object_match_error`. For binary relations the executor computes a signed margin $m_r = \mathbf{w}_r(f)^\top (p_A - p_B)$ along the dispatched axis: $\hat{r}$ (right) for LRUD, $\hat{u}$ (up) for above/below, $\hat{f}$ (forward) for front/behind. Reciprocal consistency checks that the swapped relation flips sign; the executor also evaluates a top-down projection and records `projection_inconsistency` when frame interpretations disagree.

Agent-relative, size, and metric programs. Agent-relative questions replace p_B with the camera/view origin p_0 . Clock-direction quantizes the horizontal angle of $(p_A - p_0)$ to the nearest clock sector. Size operators consume the object extent $s = (w, d, h)$ rather than the center: height uses h , width uses $\max(w, d)$, thinness $\min(w, d)$, generic size uses footprint wd or volume wdh . Metric programs report $\|p_A - p_0\|_2$ or $\|p_A - p_B\|_2$ and are gated by the conservative-numeric policy because small calibration errors flip strict numeric evaluation.

Operator-specific trust score. The verifier emits an *operator-specific trust score* as the conservative minimum over its inputs,

$$\tau = \min(\phi_{\text{lex}}(a_1), \phi_{\text{lex}}(a_2), \rho_{\min}, q_{\text{mar}}), \quad (10)$$

where $\rho_{\min} = \min_j \rho_{i_j^*}$ is the minimum quality over bound objects, q_{mar} is the bounded margin strength defined in §3.4, and object–object relations additionally fold in a projection-agreement factor $0.85 + 0.15 q_{\text{proj}}$ on q_{margin} . Agent-relative, transformation, and metric operators drop q_{margin} when the operator returns a scalar rather than a signed predicate. After graph attachment, the fusion policy uses the post-graph trust $\frac{1}{2}(\tau + C(\pi, \mathcal{E}))$, which is the τ threshold consumed by Tab. 9.

Counterfactual stability. For object–object relations with an opposite operator $\bar{\omega}$, q_{cf} in Eq. 7 averages reciprocal consistency, argument-swap consistency, near-tie stability $1 - \exp(-|m_\omega|)$, and an extent-aware perturbation score $|m_\omega|/(|m_\omega| + \varepsilon)$. We set $\varepsilon = 0.05 \max(s_a^{(k)}, s_b^{(k)}, 1)$ along the dispatched axis k , clip the mean to $[0, 1]$, and use $q_{\text{cf}} = 1$ for resolved non-opposite operators.

Override rules. Projection agreement, graph evidence, and counterfactual stability are checked before the family-specific thresholds in Tab. 9. Unsupported operators are blocked by the family gate, so the rule-fusion policy never overrides when $q_{\text{op}} = 0$. All thresholds are frozen on a held-in dev split.

Condition	Decision rule
Default row	$\tau \geq 0.75$ and graph evidence ≥ 0.60
Tiny-margin predicate	defer if $ m < 0.1$
Metric/transformation	$\tau \geq 0.90$, graph evidence ≥ 0.70 , stability ≥ 0.70
Object-agent direction	$\tau \geq 0.90$, graph evidence ≥ 0.70 , stability ≥ 0.70
Size rows	$\tau \geq 0.80$, graph evidence ≥ 0.60 , margin ≥ 0.1
Unsupported / failed	keep VLM or defer

Table 9: **Override rules used by diagnostic-guided answer selection.** The default row applies to ordinary qualitative relations after execution succeeds. Metric, direction, and size rows use stricter thresholds because these families are more sensitive to scale, frame, or near-tie margins. Unsupported or failed programs never force a geometry answer; they keep the base VLM answer or defer.

Routing behavior. The policy is deliberately asymmetric: high-confidence, operator-supported geometry may override, whereas low-trust, incomplete, or frame-ambiguous cases retain the base VLM answer. Numeric and agent-relative rows use the strictest gates because scale or viewpoint errors can change the meaning of an otherwise plausible relation. Thus, τ is an operational routing score rather than a calibrated probability; it prevents unsupported program outputs from forcing a revision.

D Qualitative Traces

Figure 3 shows one compact verifier trace, and Figure 4 lists the released trace fields.

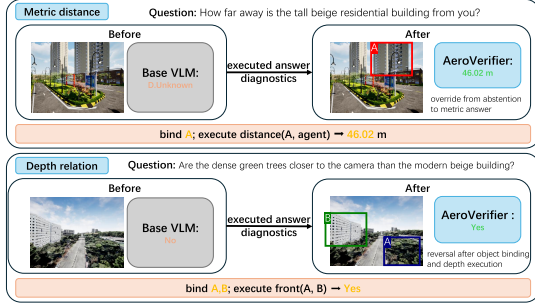


Figure 3: **Two Wuhan qualitative examples.** Each trace shows how a question is bound to object evidence, compiled into a typed spatial program, executed under the committed frame, and then routed to a final keep or override decision. The examples illustrate both the program answer and the diagnostic information exposed to the frozen VLM.

Trace fields exposed in the released JSONL. Each verifier trace records: (i) the evidence tuple used for execution; (ii) the bound object phrases and matched evidence entries; (iii) the typed program instantiated from Eq. 2; (iv) diagnostic fields such as trust, margin, and failure type; and (v) the final override/keep decision. This compact boxed summary replaces the full rendered trace panel in the appendix.

Figure 4: **Compact trace summary.** The released JSONL trace exposes the same fields used by the executor and answer-selection rule, including the evidence tuple, bound objects, typed program, trust scores, margin values, failure type, and final routing decision. This makes verifier failures auditable without rerunning the VLM.

E Implementation Details

Local Qwen2.5-VL-7B inference uses greedy decoding (`do_sample=False`, `max_new_tokens=64`) on a single RTX 4090. The GPT-5.4 API results are obtained through a commercial OpenAI-compatible API endpoint, using the provider’s `gpt-5.4` model identifier available at the time of our experiments. We treat this as a fixed API-backed proprietary VLM baseline rather than as a locally reproducible checkpoint. All API calls use deterministic settings where exposed by the service (`temperature=0`, `top_p=1`) and a fixed per-row seed; prompts, normalized outputs, and run metadata are logged for audit and will be released with the code where permitted by the API terms. Split identifiers, evidence traces, decoding settings, calibration files, and evaluation scripts will also be

released where permitted by the API and dataset terms. The verifier itself is deterministic Python with no sampling.

Reconstructed mask-depth evidence.

The reconstructed regime of §3.3 uses CIDAS/clipseg-rd64-refined for phrase grounding (a soft mask is thresholded at $\max(0.45, \mu + \sigma)$ over the probability map, with a top-quantile fallback for weak activations) and SAM vit_h box-prompt refinement (the highest predicted-*IoU* mask of the multi-mask output, falling back to the detector box when no mask is returned). The back-projection of Eq. 3 uses Open3D-VQA’s released aligned depth maps (one .npy per RGB frame, same resolution) and a pinhole intrinsic with horizontal field of view 90° , principal point at the image center, and unit depth scale; pixels that are non-finite, non-positive, or at the sentinel value (≥ 65000) are discarded, and at most 20000 valid pixels per object are kept. The robust statistics $(\mu_i, s_i, b_i, \rho_i)$ are computed by the same routine used for the provided point clouds.

E.1 Matcher Validation

On a 200-row random sample (seed 42), the deterministic `approx_correct` matcher agrees with an independent GPT-5.4 API judge on 192/200 rows (96.0% agreement; Cohen’s kappa = 0.919) and reaches 54.50% accuracy under a 25% numeric tolerance. The eight disagreements split between adjacent clock-direction false positives and uncovered yes/no synonyms; the summary is in Tab. 10.

Metric	Value
Sample size	200
Matcher accuracy	54.50%
Agreement	192/200 (96.0%)
Cohen’s κ	0.919
Disagreements	8

Table 10: **Matcher validation.** Numeric answers use a 25% relative tolerance; multiple-choice answers use exact labels.

Scope of agreement. This validation tests the scoring mechanism rather than the VLM or executor. The same matcher is used for every compared answer source, so residual boundary cases are applied consistently across methods. We retain exact matching for multiple-choice responses and reserve the numeric tolerance for metric outputs, avoiding comparisons between semantically different answer types as if they were raw strings.

F Additional Diagnostics

This appendix collects the quantitative diagnostics that support the main evidence-interface claims. The scale and corruption tables are placed in the main text because they directly support the rebuttal claims; the following tables provide supporting implementation and failure diagnostics.

F.1 Efficiency and Failure Diagnostics

The following tables report the executor efficiency profile and typed failure distribution.

Task	s/row	CUDA (GiB)	RSS (MiB)
AeroVerifier exec. (provided 3D)	0.0013	0	78
Qwen image-only	0.5449	16.58	4737
Qwen + 3D prompt	0.3783	18.61	4737
Qwen ViperGPT-style	0.4848	18.65	4737

Table 11: **Efficiency profile.** Timings exclude mask-depth perception.

Failure type	N	%
null (no failure)	751	75.1
parse_error	117	11.7
projection_inconsistency	113	11.3
object_match_error	15	1.5
frame_transformation_risk	4	0.4

Table 12: **Failure-type distribution on the balanced Open3D-VQA subset.** Most rows execute without a typed failure. The remaining failures are dominated by parser coverage and projection consistency, rather than by unsupported operators. This explains the Overall-vs-Executable gaps in Tab. 1 and motivates improving parsing and object-level evidence construction.

F.2 Corrupted-Evidence Stress Test

The complete corrupted-evidence table is reported in Tab. 7; it is kept in the main text because it directly supports the deployment-bottleneck claim.