WHEN VISION NEEDS A SECOND LOOK: TOOL-AUGMENTED ACTIVE PERCEPTION FOR EARTH OBSERVATION

Atul Dev*
Indian Institute of Technology Roorkee
atul_d@ma.iitr.ac.in

Shivank Garg*
Indian Institute of Technology Roorkee
shivank_g@mfs.iitr.ac.in

ABSTRACT

Earth Observation (EO) uses satellite and aerial imagery to monitor the Earth's surface, supporting critical applications in infrastructure, agriculture, and climate change. Current Vision-Language Models (VLMs) remain unreliable on fine-grained, numerically precise geospatial tasks. Recent evaluations on benchmarks like GeoBench-VLM highlight this shortcoming: even state-of-the-art models fail on domain-specific problems such as object counting, crop-type recognition, and vegetation-health assessment, in part because single-pass inference offers no mechanism for targeted inspection or verification. We present *GeoScout-Agent*, which augments GPT-5-mini with tool-based active perception. Built on LangChain, the agent iteratively invokes zooming/sharpening, code execution, retrieval, and vision modules (DINOv3 and SAM3) to verify intermediate hypotheses and refine predictions. Across GeoBench-VLM, *GeoScout-Agent* improves relative performance by 17.3% over the tool-free baseline.

1 INTRODUCTION

Geospatial reasoning is inherently an active and iterative process. Human analysts do not interpret satellite imagery through a single static view; instead, they dynamically zoom into regions of interest, enhance visual contrast, isolate object clusters, and cross-check local evidence before arriving at a decision. In contrast, most contemporary VLMs operate in a passive, single-pass manner; this setup breaks on small objects, artifacts, and multiscale spatial cues.

GeoBench-VLM (Danish et al., 2025) exposes this gap: even strong models like LLaVA-OneVision (Li et al., 2024) plateau around 41% accuracy and struggle with numerically precise, multiscale, or spatially grounded tasks (e.g., counting and crop classification).

We present *GeoScout-Agent*, an agentic EO framework that operationalizes active perception for geospatial reasoning. Our system integrates a base vision-language model, GPT-5-mini (OpenAI, 2025) with a LangChain (Topsakal & Akinci, 2023) controller that iteratively gathers evidence by calling EO tools (zoom/sharpen, DINOv3 (Siméoni et al., 2025), SAM3 (Carion et al., 2025), code execution, and web search) to test and refine hypotheses.

Our contributions are:

- We propose *GeoScout-Agent*, an EO agent that turns VLM inference into a closed-loop perception–action cycle.
- A tool suite for multiscale inspection and segmentation-assisted counting.
- Improved GeoBench-VLM performance, especially on dense counting and spatial reasoning.

*Equal contribution

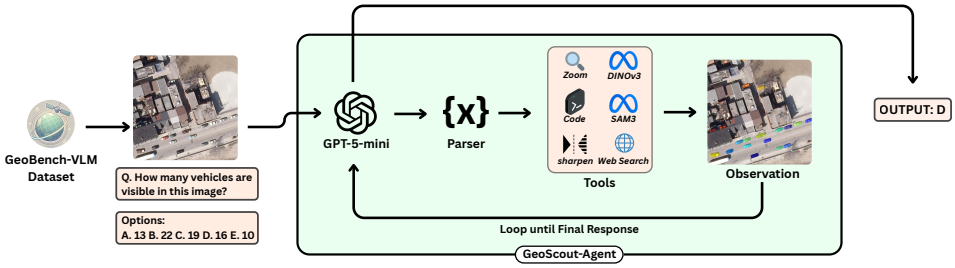


Figure 1: Framework overview of GeoScout-Agent, a tool-augmented Earth Observation (EO) agent.

2 Related Work

Prior work adapts general-purpose VLMs to earth observation via fine-tuning, dataset curation, and multi-sensor alignment, as seen in models like GeoChat (Kuckreja et al., 2024), LHRS-Bot (Muhtar et al., 2024), EarthDial (Soni et al., 2025), SkySense (Luo et al., 2024). However, these systems largely remain single-pass and struggle with numerically precise or spatially grounded tasks. In parallel, foundation models such as DINOv3 (Siméoni et al., 2025) and SAM3 (Carion et al., 2025) provide strong representations and zero-shot segmentation, yet are often used as standalone predictors or preprocessing (Xiao et al., 2025) rather than as interactive subroutines. Recent EO agents-including GeoFlow (Bhattaram et al., 2025), Remote Sensing ChatGPT (Guo et al., 2024), TreeGPT (Du et al., 2023), Change-Agent (Liu et al., 2024a), GeoMap-Agent (Huang et al., 2025) and RS-Agent (Xu et al., 2024)-orchestrate tools and retrieval, but typically emphasize knowledge retrieval and decision routing, rather than visual grounding. GeoScout-Agent instead integrates perception modules and image manipulation in a closed loop to refine intermediate hypotheses.

3 Methodology

3.1 Framework Architecture

Our framework is built upon LangChain, a widely used open-source ecosystem for constructing LLM-powered autonomous agents. It distinguishes itself by managing complex system states, integrating external tools, and facilitating prompt chaining. These capabilities are critical for moving beyond static, single-pass generation, allowing us to design a dynamic and iterative orchestration loop. Leveraging this ecosystem, we implement geospatial reasoning as a two-node graph-based agent in the LangChain ecosystem (Chase, 2022): (i) a **Reasoning Node (Agent)** that reads the current AgentState and emits tool calls or a final answer, and (ii) an **Execution Node (Tools)** that parses tool calls, runs Python routines or model inference (e.g., SAM3, DINOv3), and appends results to the state.

AgentState is an ordered message list that accumulates user inputs, agent actions, and tool outputs; retaining the full trace lets *GeoScout-Agent* reuse prior visual evidence, failed attempts, and intermediate computations instead of relying only on the latest observation. GPT-5-mini serves as the orchestrator (Figure 1): each iteration consumes AgentState, decides whether additional evidence is needed, and either invokes tools or returns an answer; tool outputs are appended for subsequent reasoning.

3.2 Visual Perception and Dynamic Analysis Tools

The agent exposes a modular toolset that can be invoked at any step, and tool-generated artifacts (e.g., masks, crops, enhanced views) are fed back into the multimodal context so the agent can reason over transformed inputs. The tool suite includes: (i) **SAM3 instance segmentation** for zero-shot instance masks with confidence scores for counting, region isolation, and pixel-level analysis, (ii) **DINOv3 unsupervised detection** for class-agnostic object discovery via dense-feature clustering and bounding-box extraction, (iii) **Python code execution** for deterministic image analysis and transformations such as pixel statistics, geometry, and custom OpenCV filters, (iv) **zoom and sharpen** for cropping around agent-selected normalized coordinates; unsharp masking to enhance

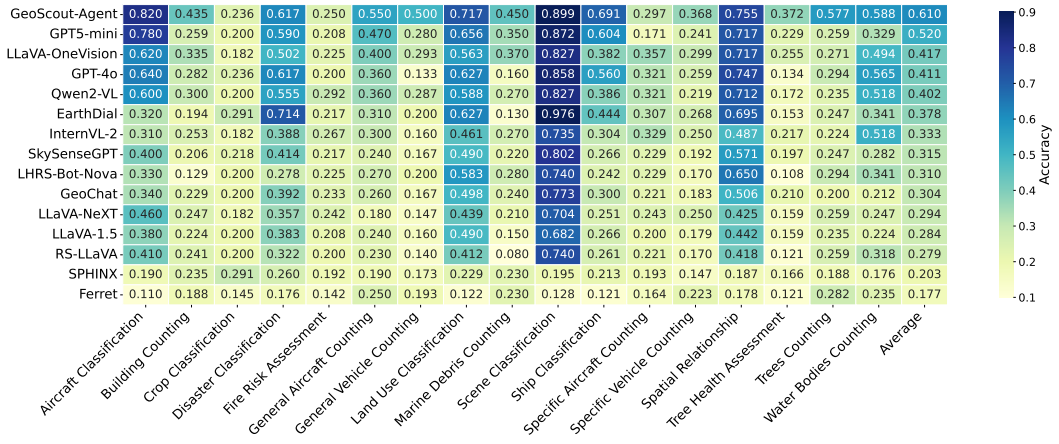


Figure 2: Task-level accuracy heatmap of VLMs on GeoBench-VLM Dataset. Tool-use outperforms single-pass models in complex visual and spatial analysis (particularly for object counting, scene understanding, and spatial reasoning).

edges in degraded imagery, and (v) **web search** for retrieval of contextual verification or fact checking when imagery alone is insufficient.

3.3 Inference Workflow

Inference proceeds through an iterative perception-reasoning LangGraph loop. At each iteration, *GeoScout-Agent* evaluates the current state, determines whether additional evidence is required, and either invokes tools or generates a final response. Tool outputs are appended to the state and incorporated into subsequent reasoning steps. The loop terminates (i) The agent produces a final answer without requesting further tools, (ii) A predefined maximum number of tool calls is reached (fixed to 3 across all experiments), or (iii) An empty response is detected.

In cases (ii) and (iii), forced synthesis is triggered, wherein the LLM is invoked without tool bindings to generate a final textual answer. The graph is additionally compiled with a recursion limit of 20 to prevent unbounded execution. These safeguards ensure bounded computation, prevent infinite loops, and guarantee that inference always yields a textual prediction.

4 Experiments and Results

We evaluate two models in addition to those in the GEOBench-VLM benchmark (Danish et al., 2025): GPT-5-mini (tool-free baseline) and *GeoScout-Agent*. Baseline scores for prior models are reproduced from GEOBench-VLM; example tasks are provided in Appendix F.

Model	Event Det	Object Cls	Counting	Scene Und	Image Cap
GPT-4o (Hurst et al., 2024)	0.4726	0.5863	0.3965	0.7114	0.6418
EarthDial (Soni et al., 2025)	**0.5418**	0.4039	0.3626	<u>0.7705</u>	0.5378
Qwen2-VL (Wang et al., 2024)	0.4640	0.4560	0.4019	0.6761	0.5895
LLaVA-OneVision (Li et al., 2024)	0.4063	0.4593	<u>0.4377</u>	0.6636	0.6317
GPT-5-mini (OpenAI, 2025)	0.4579	<u>0.6613</u>	0.3971	0.7294	<u>0.8533</u>
GeoScout-Agent	<u>0.4901</u>	**0.7330**	**0.5261**	**0.7728**	**0.8555**

Table 1: VLMs' accuracies across geospatial tasks. Evaluation includes event detection, object classification, counting, scene understanding, and image captioning. We report full results against all GeoBench-VLM baselines in the Appendix A.

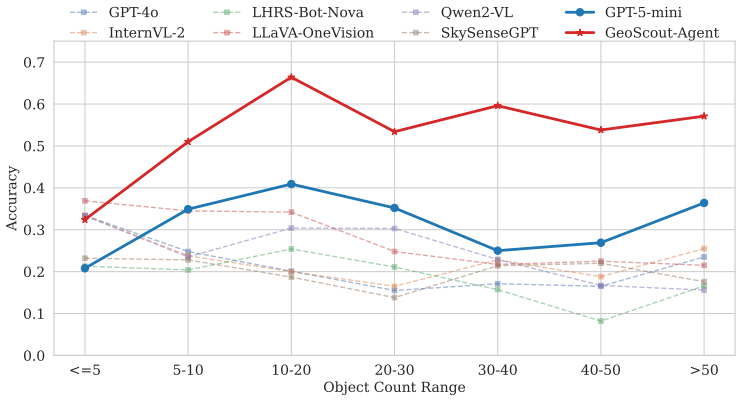


Figure 3: Object-density vs. counting accuracy

4.1 Experiments Across Task Categories

Scene Understanding: GeoScout-Agent outperforms single-pass VLMs in scene understanding (Table 1), indicating that iterative inspection and targeted refinement help resolve fine-grained scene distinctions.

Object Classification: For visually similar categories (e.g., ships and aircraft) relevant to maritime surveillance and airspace monitoring, *GeoScout-Agent* yields a 10.84% relative improvement over the baseline.

Event Detection: Although EarthDial is best in this category, *GeoScout-Agent* consistently outperforms its tool-free counterpart, suggesting iterative inspection and targeted tool use improve event-level classification.

Caption Generation: Evaluated with BERTScore, *GeoScout-Agent* provides only marginal gains, consistent with captioning emphasizing holistic semantics over fine-grained verification.

Object Localization and Counting: Counting/localization tasks (buildings, trees, water bodies, vehicles, aircraft) are multiple-choice classification problems and remain a major failure mode for monolithic VLMs. Both GPT-5-mini and *GeoScout-Agent* perform strongly (Figure 2), with the largest gains in dense scenes: building counting 67.95%, tree counting 122.78%, water body counting 78.72%, general aircraft counting 17.02%, general vehicle counting 78.57%, specific aircraft type counting 73.68%, specific vehicle type counting 52.70%, and marine debris counting 28.57%. LLaVA-OneVision remains strongest on specific aircraft counting. All improvements are relative to the corresponding tool-free baselines. Figure 3 shows accuracy versus density (sparse ≤ 5 objects to dense > 50). Most baselines degrade sharply beyond 20 objects; GPT-5-mini is more stable but still declines at high density. *GeoScout-Agent* is consistently higher across densities, with the largest gains in medium/high-density regimes, indicating that zooming, instance separation, and verification mitigate occlusion/overlap-driven errors.

4.2 Accuracy Latency Tradeoff in Tool-Augmented Geospatial Reasoning

Tool augmentation increases compute: *GeoScout-Agent* averages 4.30 seconds/sample vs. 0.90 seconds/sample tool-free, uses $1,884$ vs. 779 total tokens per interaction ($2.42\times$), and makes 2.79 tool calls/sample on average; the overhead is primarily due to persistent state, intermediate tool outputs, and iterative reasoning.

5 Conclusion

Earth observation requires reasoning beyond passive recognition: many GeoBench-VLM failures reflect single-pass inference limits rather than model capacity. We introduce *GeoScout-Agent*, a GPT-5-mini-based, tool-augmented agent that enables active perception via iterative inspection, local refinement, and explicit verification using segmentation, detection, zooming, sharpening, and

lightweight code execution. Across GeoBench-VLM, this agentic formulation consistently improves performance, with the largest gains in object counting, scene understanding and object classification.

6 LIMITATIONS

Our framework reflects practical resource constraints: iterative tool use increases token and compute cost, so we focus on cost-efficient models and run a single evaluation per configuration (no repeated runs or extensive ablations). Results should therefore be interpreted primarily as comparative trends rather than run-to-run variability.

ACKNOWLEDGEMENTS

We thank the reviewers and organizers for their valuable feedback. This work was supported in part by compute credits from Lambda Labs.

REFERENCES

Yakoub Bazi, Laila Bashmal, Mohamad Mahmoud Al Rahhal, Riccardo Ricci, and Farid Melgani. Rs-llava: A large vision-language model for joint captioning and question answering in remote sensing imagery. *Remote Sensing*, 16(9):1477, 2024.

Amulya Bhattaram, Justin Chung, Stanley Chung, Ranit Gupta, Janani Ramamoorthy, Kartikeya Gullapalli, Diana Marculescu, and Dimitrios Stamoulis. Geoflow: Agentic workflow automation for geospatial tasks, 2025. URL https://arxiv.org/abs/2508.04719.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. Sam 3: Segment anything with concepts. *arXiv preprint arXiv:2511.16719*, 2025.

Harrison Chase. Langchain. https://github.com/langchain-ai/langchain, 2022.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 24185–24198, 2024.

Muhammad Danish, Muhammad Akhtar Munir, Syed Roshaan Ali Shah, Kartik Kuckreja, Fahad Shahbaz Khan, Paolo Fraccaro, Alexandre Lacoste, and Salman Khan. Geobench-vlm: Benchmarking vision-language models for geospatial tasks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7132–7142, 2025.

Siqi Du, Shengjun Tang, Weixi Wang, Xiaoming Li, and Renzhong Guo. Tree-gpt: Modular large language model expert system for forest remote sensing image understanding and interactive analysis, 2023. URL https://arxiv.org/abs/2310.04698.

Haonan Guo, Xin Su, Chen Wu, Bo Du, Liangpei Zhang, and Deren Li. Remote sensing chatgpt: Solving remote sensing tasks with chatgpt and visual models, 2024. URL https://arxiv.org/abs/2401.09083.

Yangyu Huang, Tianyi Gao, Haoran Xu, Qihao Zhao, Yang Song, Zhipeng Gui, Tengchao Lv, Hao Chen, Lei Cui, Scarlett Li, and Furu Wei. Peace: Empowering geologic map holistic understanding with mllms, 2025. URL https://arxiv.org/abs/2501.06184.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Kartik Kuckreja, Muhammad Sohail Danish, Muzammal Naseer, Abhijit Das, Salman Khan, and Fahad Shahbaz Khan. Geochat: Grounded large vision-language model for remote sensing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 27831–27840, 2024.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024.

Zhenshi Li, Dilxat Muhtar, Feng Gu, Yanglangxing He, Xueliang Zhang, Pengfeng Xiao, Guangjun He, and Xiaoxiang Zhu. Lhrs-bot-nova: Improved multimodal large language model for remote sensing vision-language interpretation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 227:539–550, 2025.

Ziyi Lin, Chris Liu, Renrui Zhang, Peng Gao, Longtian Qiu, Han Xiao, Han Qiu, Chen Lin, Wenqi Shao, Keqin Chen, et al. Sphinx: The joint mixing of weights, tasks, and visual embeddings for multi-modal large language models. *arXiv preprint arXiv:2311.07575*, 2023.

Chenyang Liu, Keyan Chen, Haotian Zhang, Zipeng Qi, Zhengxia Zou, and Zhenwei Shi. Change-agent: Towards interactive comprehensive remote sensing change interpretation and analysis, 2024a. URL https://arxiv.org/abs/2403.19646.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023. URL https://arxiv.org/abs/2304.08485.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llavanext: Improved reasoning, ocr, and world knowledge, 2024b.

Junwei Luo, Zhen Pang, Yongjun Zhang, Tingzhu Wang, Linlin Wang, Bo Dang, Jiangwei Lao, Jian Wang, Jingdong Chen, Yihua Tan, and Yansheng Li. Skysensegpt: A fine-grained instruction tuning dataset and model for remote sensing vision-language understanding, 2024. URL https://arxiv.org/abs/2406.10100.

Dilxat Muhtar, Zhenshi Li, Feng Gu, Xueliang Zhang, and Pengfeng Xiao. Lhrs-bot: Empowering remote sensing with vgi-enhanced large multimodal language model, 2024. URL https://arxiv.org/abs/2402.02544.

OpenAI. Gpt-5-mini model documentation. https://platform.openai.com/docs/models, 2025. Accessed: 2025-11-30.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

Sagar Soni, Akshay Dudhane, Hiyam Debary, Mustansar Fiaz, Muhammad Akhtar Munir, Muhammad Sohail Danish, Paolo Fraccaro, Campbell D Watson, Levente J Klein, Fahad Shahbaz Khan, et al. Earthdial: Turning multi-sensory earth observations to interactive dialogues. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 14303–14313, 2025.

Oguzhan Topsakal and Tahir Cetin Akinci. Creating large language model applications utilizing langchain: A primer on developing llm apps fast. In *International conference on applied engineering and natural sciences*, volume 1, pp. 1050–1056, 2023.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution, 2024. URL https://arxiv.org/abs/2409.12191.

Aoran Xiao, Weihao Xuan, Junjue Wang, Jiaxing Huang, Dacheng Tao, Shijian Lu, and Naoto Yokoya. Foundation models for remote sensing and earth observation: A survey, 2025. URL https://arxiv.org/abs/2410.16602.

Wenjia Xu, Zijian Yu, Boyang Mu, Zhiwei Wei, Yuanben Zhang, Guangzuo Li, and Mugen Peng. Rs-agent: Automating remote sensing tasks through intelligent agent. *arXiv preprint arXiv:2406.07089*, 2024.

Haoxuan You, Haotian Zhang, Zhe Gan, Xianzhi Du, Bowen Zhang, Zirui Wang, Liangliang Cao, Shih-Fu Chang, and Yinfei Yang. Ferret: Refer and ground anything anywhere at any granularity. *arXiv preprint arXiv:2310.07704*, 2023.

Appendix

A Additional Experimental Results

In this section, we provide extended evaluations and ablation studies that were omitted from the main text due to space constraints.

A.1 Referring Expression Detection

This is a visual grounding task where the model must localize objects in aerial/satellite imagery based on natural language referring expressions. Unlike multiple-choice QA, the model outputs bounding box coordinates rather than selecting an answer. GPT-5-mini shows lower performance on this task, whereas Sphinx achieves the highest precision at both IoU thresholds (Table 2). Nevertheless, *GeoScout-Agent* consistently improves GPT-5-mini's precision, indicating that explicit segmentation and region-level reasoning partially mitigate localization challenges, even in tasks where overall performance remains limited.

Figure 4 presents referring expression detection performance across object sizes. GPT-5-mini performs poorly across all regimes, revealing limitations in fine-grained localization. Tool augmentation offers only marginal gains, particularly for large objects, suggesting that failures primarily arise from representational and grounding limitations rather than insufficient visual resolution.

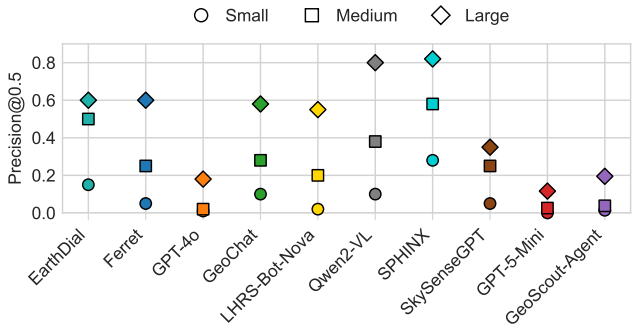


Figure 4: Performance comparison of different models on Referring Expression Detection across various object sizes

Model	Prec@0.5	Prec@0.25
Sphinx (Lin et al., 2023)	**0.3408**	**0.5289**
EarthDial (Soni et al., 2025)	0.2429	0.4139
GeoChat (Kuckreja et al., 2024)	0.1151	0.2100
Ferret (You et al., 2023)	0.0943	0.2003
Qwen2-VL (Wang et al., 2024)	0.1518	0.2524
GPT-4o (Hurst et al., 2024)	0.0087	0.0386
LHRS-Bot-Nova (Li et al., 2025)	0.0930	0.2423
SkySenseGPT (Luo et al., 2024)	0.1082	0.3224
GPT-5-mini (OpenAI, 2025)	0.0491	0.1631
GeoScout-Agent	0.0756	0.2204

Table 2: Referring expression detection results. We report Precision at IoU thresholds 0.5 and 0.25.

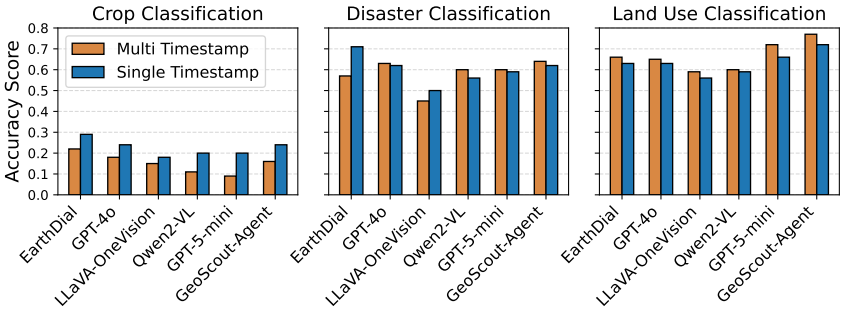


Figure 5: Single- versus multi-temporal performance comparison across crop classification, disaster classification, and land use classification tasks.

Model	Crop Cls	Damaged Bldg Cnt	Disaster Cls	Farm Pond CD	Land Use Cls
EarthDial (Soni et al., 2025)	**0.2182**	0.4362	0.5727	<u>0.2105</u>	0.6623
GPT-4o (Hurst et al., 2024)	<u>0.1818</u>	**0.5667**	<u>0.6300</u>	0.1711	0.6525
LLaVA-OneVision (Li et al., 2024)	0.1455	0.4810	0.4537	0.1842	0.5869
Qwen2-VL (Wang et al., 2024)	0.1091	<u>0.5000</u>	0.5991	0.1974	0.5967
GPT-5-mini (OpenAI, 2025)	0.0909	0.3095	0.5991	0.1600	<u>0.7180</u>
GeoScout-Agent	0.1636	0.4459	**0.6388**	**0.2500**	**0.7705**

Table 3: Performance of vision-language models on temporal geospatial tasks. Evaluation covers crop classification, damaged building counting, disaster type classification, farm pond change detection (CD), and land use classification. Results show that GeoScout-Agent consistently improves GPT-5-mini across all temporal tasks.

A.2 Temporal Understanding

Temporal reasoning tasks evaluate changes across time, which are central to remote sensing analysis. We evaluate five temporal tasks: crop classification, damaged building counting, disaster type classification, farm pond change detection, and land use classification (Table 3). *GeoScout-Agent* consistently improves over GPT-5-mini across all tasks, highlighting the benefit of tool-assisted reasoning for modeling temporal changes.

Despite these gains, absolute performance remains low for crop classification and farm pond change detection, indicating persistent challenges in capturing long-term temporal dependencies. These results suggest that current VLM-based approaches are insufficient for complex temporal reasoning and motivate future work on stronger temporal modeling mechanisms.

Figure 5 compares single- and multi-temporal performance for crop, disaster, and land use classification. Multi-temporal inputs reduce accuracy for crop classification across most models, including GPT-5-mini, indicating that temporal variability introduces noise without explicit alignment. In contrast, disaster classification benefits from pre- and post-event observations, while land use classification shows consistent gains due to temporal stability. When combined, multi-temporal inputs and *GeoScout-Agent* yield compounding improvements for temporally informative tasks.

Model	Event Det	Object Cls	Counting	Scene Und	Image Cap
GPT-4o (Hurst et al., 2024)	0.4726	0.5863	0.3965	0.7114	0.6418
EarthDial (Soni et al., 2025)	**0.5418**	0.4039	0.3626	<u>0.7705</u>	0.5378
Qwen2-VL (Wang et al., 2024)	0.4640	0.4560	0.4019	0.6761	0.5895
LLaVA-OneVision (Li et al., 2024)	0.4063	0.4593	<u>0.4377</u>	0.6636	0.6317
SkySenseGPT (Luo et al., 2024)	0.3458	0.3094	0.3119	0.6205	0.6416
InternVL-2 (Chen et al., 2024)	0.3458	0.3062	0.3280	0.5727	0.5968
GeoChat (Kuckreja et al., 2024)	0.3372	0.3127	0.2922	0.6091	0.4395
LHRS-Bot-Nova (Li et al., 2025)	0.2594	0.2704	0.3286	0.6330	0.6275
LLaVA-NeXT (Liu et al., 2024b)	0.3170	0.3192	0.2737	0.5477	0.6293
LLaVA-1.5 (Liu et al., 2023)	0.3228	0.3029	0.2618	0.5625	0.6346
RS-LLaVA (Bazi et al., 2024)	0.2795	0.3094	0.2534	0.5534	0.5604
SPHINX (Lin et al., 2023)	0.2363	0.2052	0.1860	0.2170	0.6451
Ferret (You et al., 2023)	0.1643	0.1173	0.1956	0.1261	0.5615
GPT-5-mini (OpenAI, 2025)	0.4579	<u>0.6613</u>	0.3971	0.7294	<u>0.8533</u>
GeoScout-Agent	<u>0.4901</u>	**0.7330**	**0.5261**	**0.7728**	**0.8555**

Table 4: VLMs accuracies across geospatial tasks. Evaluation includes event detection, object classification, counting, scene understanding, and image captioning.

A.3 Full Results Table

Table 4 showcases complete results on the geospatial tasks referenced in the main text.

B Baseline Performance of Individual Tools

To analyze the contribution of individual tools, we report baselines with *GeoScout-Agent*'s tools enabled in isolation. We evaluate vision-only, segmentation-only, and code-execution settings, each without coordinated tool use or multi-step reasoning. These baselines reveal the limitations of single-module approaches and show that performance gains in the main results stem from integrating complementary tools rather than any individual component.

B.1 DINOv3 Baseline

DINOv3 provides unsupervised, class-agnostic object detection by leveraging self-supervised vision transformer representations. The module extracts dense patch-level features from a DINOv3 checkpoint pretrained on satellite imagery (DINOv3-ViT-L/16, 493M parameters[1]).

Images are resized so that the shortest edge is 518 pixels before passing through the backbone, and the resulting patch tokens are reshaped into a spatial grid of feature vectors (patch size of 14×14 pixels per patch). We perform unsupervised region discovery by L_2-normalizing these patch features and clustering them with K-means (fixed $K = 4$ clusters). The background cluster is identified heuristically as the cluster containing the largest number of patches, under the assumption that background regions dominate the image spatially. The remaining $K - 1$ clusters are treated as candidate object regions, we generate binary masks, upsample them to the original image resolution, extract contours, and compute bounding rectangles.

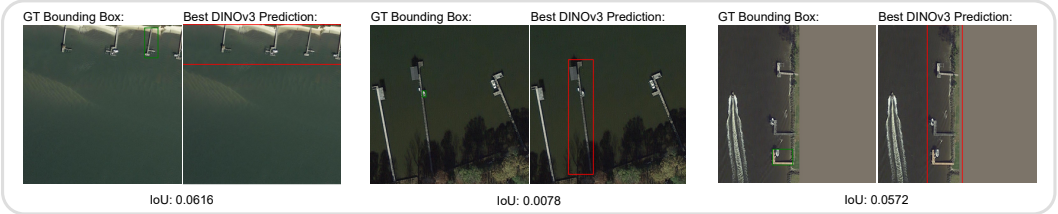


Figure 6: DINOv3 Failure Cases on GEO-bench Ref-Det. Representative low-scoring examples from unsupervised object detection using DINOv3 feature clustering. Green boxes indicate ground truth annotations. DINOv3 produces multiple candidate bounding boxes per image; for clarity, only the predicted box with the highest IoU to the ground truth (i.e., the best-matched prediction) is shown in red. Despite selecting the best match, the low IoU scores (0.0078-0.0616) demonstrate the model's difficulty with small object localization and fine-grained spatial reasoning in remote sensing imagery.

For referring expression detection, evaluation is performed using standard intersection-over-union (IoU) based metrics. The detector produces multiple candidate bounding boxes per image. For each ground-truth box, IoU is computed against all predicted boxes, and the maximum IoU is retained as the score for that ground truth. Mean IoU is then reported as the average of these per-ground-truth maximum IoUs. Using this protocol, the DINOv3-based baseline achieves a mean IoU of 8.5%. Table 5 reports the DINOv3-based baseline precision at different IoU thresholds for referring expression detection. Representative failure cases are shown in Fig 6, highlighting the limitations of unsupervised clustering-based detection on fine-grained localization tasks.

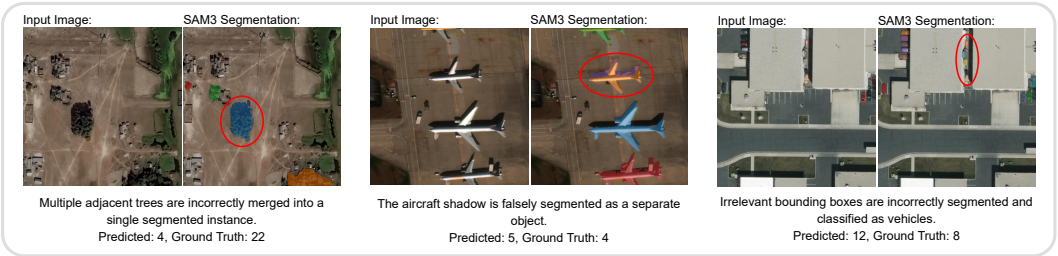


Figure 7: Representative failure cases of the SAM3 baseline on Counting Tasks, illustrating merged instances, shadow-induced false positives, and spurious object segmentation.

Metric	Prec@0.25	Prec@0.50	Prec@0.75
Precision (%)	9.62	4.37	0.15

Table 5: DINOv3 baseline performance on referring expression detection.

B.2 SAM3 Baseline

SAM3[2] is a foundation segmentation model that produces high-quality, zero-shot object masks across diverse visual domains without task-specific training.

To compute the SAM3 baseline, we apply SAM3 to all counting tasks in the single MCQ subset. For each task, the object category is extracted from the question

[1] https://huggingface.co/facebook/DINOv3-vitl16-pretrain-sat493m
[2] https://huggingface.co/facebook/sam3

Task	Correct	Total	Accuracy (%)
Building Counting	44	170	25.88
General Aircraft Counting	55	100	55.00
General Vehicle Counting	81	150	54.00
Marine Debris Counting	17	100	17.00
Specific Aircraft Type Counting	38	140	27.14
Specific Vehicle Type Counting	23	224	10.27
Trees Counting	8	85	9.41
Water Bodies Counting	43	85	50.59
Overall	309	1054	29.32

Table 6: SAM3 baseline performance on counting tasks.

prompt and provided as text input to SAM3 (e.g., "How many pickup trucks are there in the image?" $\rightarrow$ *pickup trucks*). SAM3 produces segmentation masks for the queried category, and the total number of segmented instances is taken as the predicted count. For multiple-choice evaluation, the option closest to this predicted count is selected; if two options are equally distant, one is chosen at random. The resulting per-task and overall performance is summarized in Table 6. Fig 7 presents representative failure cases of the SAM3 baseline across different scenes. These cases illustrate the limitations of single-step segmentation when applied to dense or visually ambiguous remote-sensing imagery.

Task	Correct	Total	Accuracy (%)
Building Counting	13	170	7.65
General Aircraft Counting	6	100	6.00
General Vehicle Counting	16	150	10.67
Marine Debris Counting	21	100	21.00
Specific Aircraft Type Counting	5	140	3.57
Specific Vehicle Type Counting	11	224	4.91
Trees Counting	4	85	4.71
Water Bodies Counting	8	85	9.41
Overall	84	1054	7.97

Table 7: Code execution baseline performance on counting tasks.

B.3 Code Execution Tool Baseline

For the code-execution baseline, GPT-5-mini is prompted to generate deterministic Python code for each of the eight counting tasks: Building, General Aircraft, General Vehicle, Marine Debris, Specific Aircraft Type, Specific Vehicle Type, Trees, and Water Bodies. The prompt template instructs the model to write code that processes a PIL Image object and returns an integer count. If the generated code raises an exception during execution, the model is re-prompted with the error traceback appended to the original prompt. Each counting task is evaluated independently using a task-specific prompt. The generated code is executed in a constrained environment, and the predicted integer is mapped to a multiple-choice answer by selecting the closest option; if two options are equidistant, one is chosen at random. Table 7 reports task-wise results for the code-execution baseline.

Prompting Details

All code-execution prompts follow the same structure, differing only in the target object category and task-specific query examples:

Code Execution Prompt Template (Tool Baseline)

You are an expert computer vision engineer.
Task: Given an input image containing {OBJECT}, write Python code that counts the total number of {OBJECT} present in the image.
The code should be able to answer questions like: {TASK-SPECIFIC QUERY VARIANTS}
Constraints:
- The image is already loaded as a PIL Image object named 'image'.
- You may use: PIL, NumPy, OpenCV and other standard Python libraries.
- The function must be deterministic.
- The output must be a single integer representing the number of buildings/structures in the image.
Output requirements of code:
- Return ONLY the integer count.
- No print statements.
- No explanations.
- No comments outside the code.
- Output only valid Python code.

In this template, `OBJECT` denotes the target object category specific to each counting task (e.g., trees, water bodies, vehicles), and `TASK-SPECIFIC-QUERY-VARIANTS` comprises all unique question formulations from the single-answer subset that pertain to the given task, providing the model with representative examples of expected queries.

C Hyperparameters Settings and Implementation Details

All experiments were conducted on a single NVIDIA A6000 GPU. The hyperparameters listed in Table 8 are fixed and shared across all tasks.

Component	Parameter	Value
Language Model	Model	gpt-5-mini
Language Model	Temperature	0.3
SAM3 Segmentation	Threshold	0.5
DINOv3 Detection	Feature dim	518
DINOv3 Detection	Patch size	14
DINOv3 Detection	#Clusters	4
DINOv3 Detection	Box threshold	0.02
DINOv3 Detection	Random state	42
Image Tools	Sharpen intensity Default	2.0
Image Tools	Sharpen radius Default	2
Image Tools	Sharpen threshold Default	3
Image Tools	Zoom factor Default	2.0
Image Tools	Output size	448

Table 8: Hyperparameters used across all experiments.

D IMAGE PROCESSING TOOLS

All image operations are implemented using the Pillow library.[3]

D.1 IMAGE SHARPENING TOOL

The sharpening tool applies an unsharp mask filter to enhance blurred or low-quality images, revealing fine details such as text or object boundaries. The intensity parameter controls sharpening strength, where higher values produce more pronounced edge enhancement.

Python (sharpen_image_tool)

```python
def sharpen_image_tool(image_path: str, intensity: float = 2.0) -> str:
    img = Image.open(image_path)

    sharpened_img = img.filter(
        ImageFilter.UnsharpMask(
            radius=2,
            percent=int(intensity * 100),
            threshold=3,
        )
    )

    output_path = f"{base}_sharpened_{intensity}{ext}"
    sharpened_img.save(output_path)
    return output_path
```

D.2 IMAGE ZOOM TOOL

The zoom tool extracts and magnifies a region of interest specified by normalized coordinates $(x, y) \in [0, 1]^2$, where $(0.5, 0.5)$ represents the image center. The cropped region is resized to 448×448 pixels using Lanczos interpolation.

Python (zoom_image_tool)

```python
def zoom_image_tool(
    image_path: str, x: float, y: float, zoom_factor: float = 2.0
) -> str:
    img = Image.open(image_path)
    w, h = img.size

    # Convert normalized coordinates to pixels
    center_x = int(x * w) if x <= 1.0 else int(x)
    center_y = int(y * h) if y <= 1.0 else int(y)

    # Calculate crop dimensions
    crop_w = max(1, int(w / zoom_factor))
    crop_h = max(1, int(h / zoom_factor))

    # Define crop box
    left = max(0, center_x - crop_w // 2)
    top = max(0, center_y - crop_h // 2)
    right = min(w, left + crop_w)
```

[3] https://pillow.readthedocs.io

```
        bottom = min(h, top + crop_h)

        # Crop and resize
        cropped = img.crop((left, top, right, bottom))
        zoomed_img = cropped.resize((448, 448), Image.LANCZOS)

        output_path = f"{base}_zoomed_{zoom_factor}x_{x}_{y}{ext}"
        zoomed_img.save(output_path)
        return output_path
```

E PROMPT TEMPLATES

We provide the complete prompt templates used in our evaluation framework. All prompts were designed to elicit structured reasoning and consistent answer formatting across experimental conditions.

E.1 BASELINE SYSTEM PROMPT

The following system prompt defines the tool-augmented VQA agent's capabilities and instructions for multi-turn reasoning.

Single MCQ Task

You are an expert Visual Question Answering assistant. Analyze images carefully and provide accurate answers based on what you can see.
Question: $\{question\}$ Options: $\{options\_text\}$
Please analyze the provided image carefully. Follow these steps:
1. First, explain your reasoning about what you observe in the image
2. Then, select the best option from the list above
3. Format your response as: Reason: [Your detailed analysis and reasoning] Answer: [Single letter A, B, C, D, or E]

Captioning Task

You are an expert image captioning assistant specializing in aerial and satellite imagery analysis. Task: $\{prompt\}$ Analyze the provided image and generate a detailed caption.
Focus on:
- Main objects and structures visible
- Spatial relationships between elements
- Colors, sizes, and notable features
- Overall scene context and setting
Provide a comprehensive caption that captures all important details visible in the image.

Referring Expression Detection Task

You are an expert at detecting and localizing objects in satellite/aerial images. Given a referring expression, output the bounding box coordinates of the described object(s).
Output ONLY coordinates in [[x1, y1, x2, y2], ...] format.

Referring Expression: $\{prompt\_text\}$
Image size: $\{image\_size\} \times \{image\_size\}$ pixels
IMPORTANT: The number at the beginning of the referring expression indicates EXACTLY how many objects to detect. In this case, you must output EXACTLY $\{num\_objects\}$ bounding box(es).
Locate the object(s) described by the referring expression in the image.
Output ONLY the bounding box coordinates in this exact format: [[x1, y1, x2, y2]]
Where x1,y1 is top-left corner and x2,y2 is bottom-right corner.
For multiple objects, use: [[x1, y1, x2, y2], [x1, y1, x2, y2], ...]
Coordinates must be integers in pixel values (0 to $\{image\_size\}$). Output EXACTLY $\{num\_objects\}$ box(es), nothing more, nothing less.

Temporal Task

You are an expert Visual Question Answering assistant specializing in analyzing temporal changes in satellite/aerial imagery. Analyze images carefully and provide accurate answers based on what you can see.
You are provided with $\{num\_images\}$ images showing temporal changes.
Question:$\{question\}$ Options: $\{options\_text\}$
Please analyze the provided images carefully, comparing the temporal changes.
Follow these steps:
1. First, explain your reasoning about what you observe across the images
2. Then, select the best option from the list above
3. Format your response as: Reason: [Your detailed analysis and reasoning] Answer: [Single letter A, B, C, D, or E]

E.2 Agent System Prompt

The following system prompt defines the tool-augmented VQA agent's capabilities and instructions for multi-turn reasoning.

Main System Prompt (Agent with Tools)

You are an expert Visual Question Answering (VQA) assistant with access to powerful image analysis and web search tools. Your goal is to provide accurate answers to questions about images.
AVAILABLE TOOLS AND HOW TO USE THEM:
1. **sharpen_image_tool(image_path: str, intensity: float = 2.0) -> str**
- Purpose: Enhance blurred or low-quality images to reveal fine details
- Input: image_path (file path), intensity (1.0-3.0, higher = more sharpening)
- Output: Path to the sharpened image
- When to use: Image is blurry, text is hard to read, or fine details are unclear
- Performance: Improves clarity of edges, text, and small features
2. **zoom_image_tool(image_path: str, x: float, y: float, zoom_factor: float = 2.0) -> str**
- Purpose: Magnify a specific region of an image for detailed inspection
- Input: image_path, x/y coordinates (0.0-1.0 as ratio of width/height), zoom_factor
- Output: Path to the zoomed/cropped image

- When to use: Need to examine small objects, read distant text, or analyze a specific area
- Performance: Helps identify small objects, count items in dense areas, read fine text
3. **execut_code_tool(image_path: str, code_string: str) -> str**
- Purpose: Run ANY Python code to analyze OR manipulate images
- Input: image_path, Python code (variable 'image' is PIL Image object, output to 'result')
- Output: Path to the transformed/manipulated image
- When to use:
* ANALYSIS: Extract features, count pixels, measure properties, detect patterns
* MANIPULATION: Adjust brightness/contrast, apply filters, rotate, crop, color adjustments
- Performance: Extremely flexible - can do anything Python/PIL/OpenCV supports
- IMPORTANT: If you manipulate an image, the output path can be used in subsequent tool calls!
- Examples:
* Analysis: "import numpy as np; hist = np.histogram(np.array(image)); result = image"
* Manipulation: "from PIL import ImageEnhance; result = ImageEnhance.Contrast(image).enhance(2.0)" * Custom filter: "result = image.filter (ImageFilter.EDGE_ENHANCE_MORE)"
4. **sam3(image_path: str, prompt: str) -> str**
- Purpose: Segment and highlight specific objects using text descriptions
- Input: image_path, prompt (object description like "cars", "buildings", "trees")
- Output: Path to image with colored segmentation masks overlaid
- When to use: Count objects, identify object locations, analyze object distribution
- Performance: Excellent for object counting, spatial analysis, and object identification
5. **DINOv3(image_path: str) -> str**
- Purpose: Automatically detect and box salient objects without labels
- Input: image_path
- Output: Path to image with bounding boxes around detected objects
- When to use: Discover unknown objects, find regions of interest, unsupervised object detection
- Performance: Good for finding distinct regions, objects, or structures without prior knowledge
IMPORTANT INSTRUCTIONS:
- ALWAYS analyze the question first to determine which tools might help
- Use tools EFFICIENTLY - you have a maximum of 3 tool calls, so choose wisely
- You can use MULTIPLE tools in sequence (e.g., sharpen $\rightarrow$ zoom $\rightarrow$ sam3), but be strategic
- For counting tasks, use sam3 or DINOv3
- For reading small text, use sharpen_image_tool or zoom_image_tool
- The conversation history shows all previous tool outputs - use them for your final answer

- After using tools, you MUST provide a clear, concise text answer based on ALL information gathered - If the image alone is sufficient, you may answer directly without tools
MULTI-TURN STRATEGY:
- First turn: Use necessary tools to gather information
- Subsequent turns: Analyze tool outputs and refine if needed
- Final turn: YOU MUST provide a TEXT response with your definitive answer
CRITICAL: After using tools, you MUST provide a final text answer. Do not end the conversation with just tool calls - always conclude with your reasoning and answer in text format.
Your responses should be accurate, well-reasoned, and based on the evidence from the image and tool outputs.

Referring Expression Detection: Agent System Prompt

You are an expert Visual Question Answering (VQA) assistant specialized in detecting and localizing objects in satellite/aerial images based on referring expressions.
YOUR TASK:
Given an image and a referring expression (e.g., "1 baseball-diamond at the bottom"), you must identify and output the bounding box coordinates of the described object(s).
OUTPUT FORMAT - CRITICAL:
You MUST output your answer as bounding box coordinates in the following JSON format: [[x1, y1, x2, y2], [x1, y1, x2, y2], ...]
Where:
- x1, y1 = top-left corner coordinates
- x2, y2 = bottom-right corner coordinates
- Multiple boxes are separated by commas within the outer list
- Coordinates are in pixels (integer values)
EXAMPLE OUTPUTS:
- Single object: [[420, 556, 891, 1021]]
- Two objects: [[420, 556, 891, 1021], [100, 200, 300, 400]]
AVAILABLE TOOLS AND WHEN TO USE THEM:
1. **sharpen_image_tool** - Enhance blurry images to see details better
2. **zoom_image_tool** - Magnify specific regions for detailed inspection
3. **execute_code_tool** - Run custom Python code for image analysis
4. **sam3** - Segment objects using text prompts (useful for finding object boundaries)
5. **DINOv3** - Detect salient objects automatically (useful for discovering object locations)
IMPORTANT INSTRUCTIONS: - Analyze the image carefully to locate the described object(s)
- Pay attention to spatial references (top, bottom, left, right, center)
- Count the number of objects mentioned in the expression
- Provide coordinates that tightly bound the described object(s)
- Use tools if needed to better identify object locations
- Your final answer MUST be in the JSON bounding box format specified above

- Do NOT include any other text in your final answer - ONLY the coordinate list
CRITICAL: Your final response must be ONLY the bounding box coordinates in the format [[x1, y1, x2, y2], ...]. No explanations, no reasoning - just the coordinates.

The temporal and captioning tasks share the same agent system prompt as the primary agent, with no task-specific modifications.

E.3 Agent Query Format

The agent prompt additionally includes the image file path for tool invocation and encourages strategic tool use.

Single MCQ Task

Image file path: $\{img\_path\}$
Question: $\{question\}$ Options: $options\_text$
Analyze the provided image and answer the question by selecting the best option from the list above.
Consider using your available tools if they can help improve accuracy.
IMPORTANT: When calling tools, use the exact image path provided above: $\{img\_path\}$
After gathering information (with or without tools), provide your final response in this format:
Reason: [Your detailed analysis and reasoning based on the image and any tool outputs] Answer: [Single letter A, B, C, D, or E]

Captioning Task

Image file path: $\{image\_path\}$
Task: $\{prompt\}$
Analyze the provided image and generate a detailed caption. You may use your available tools to:
- Segment specific objects (sam3) to identify and count elements
- Detect objects automatically (DINOv3) to find structures
- Zoom into specific areas for detail
- Sharpen the image if needed
IMPORTANT: When calling tools, use the exact image path: $\{image\_path\}$
After analyzing the image (with or without tools), provide your final caption that describes:
- Main objects and structures visible
- Spatial relationships between elements
- Notable features, colors, and sizes
- Overall scene context

Referring Expression Detection Task

Image file path: $\{image\_path\}$
Image size: $\{image\_size\} \times \{image\_size\}$ pixels
Referring Expression: $\{prompt\_text\}$

IMPORTANT: The number at the beginning of the referring expression indicates EXACTLY how many objects to detect.
In this case, you must output EXACTLY $\{num\_objects\}$ bounding box(es).
Locate the object(s) described by the referring expression.
You may use available tools (sam3, DINOv3, zoom, etc.) to help identify the object location. IMPORTANT: When calling tools, use the exact image path: $\{image\_path\}$
After analysis, output ONLY the bounding box coordinates in this exact format:
[[x1, y1, x2, y2]]
Where x1,y1 is top-left corner and x2,y2 is bottom-right corner.
For multiple objects: [[x1, y1, x2, y2], [x1, y1, x2, y2], ...]
Coordinates must be integers in pixel values (0 to $\{image\_size\}$). Output EXACTLY $\{num\_objects\}$ box(es), nothing more, nothing less.

Temporal Task

You are provided with $\{num\_images\}$ images showing temporal changes. Image file paths:
$\{image\_paths\_str\}$
Question: $\{question\}$ Options: $\{options\_text\}$
Analyze the provided images (comparing temporal changes) and answer the question by selecting the best option from the list above.
Consider using your available tools if they can help improve accuracy.
IMPORTANT: When calling tools, use the exact image paths provided above.
After gathering information (with or without tools), provide your final response in this format:
Reason: [Your detailed analysis and reasoning based on the images and any tool outputs]
Answer: [Single letter A, B, C, D, or E]

E.4 Agent Loop Termination Prompt

When the tool-call budget is exhausted, forcing reminders are applied to guarantee answer generation; no reminder is needed for captioning, and the temporal task follows the same reminder as the single-step MCQ task.

Single MCQ Task

You have gathered sufficient information from tools. Now provide your final answer as ONLY a single letter (A, B, C, D, or E). Do NOT call any more tools.

Referring Expression Detection Task

You have gathered sufficient information from tools. Now provide your final answer as ONLY the bounding box coordinates in the format [[x1, y1, x2, y2], ...]. Do NOT call any more tools. Do NOT include any explanations.

F GeoBench-VLM Dataset Examples

We present illustrative examples from the GeoBench-VLM dataset (Fig 8). These samples highlight the diverse range of remote sensing tasks, categorized into Scene Understanding, Object Classification, Localization & Counting, Event Detection, Caption Generation, Temporal Understanding and Referring Expression Detection.

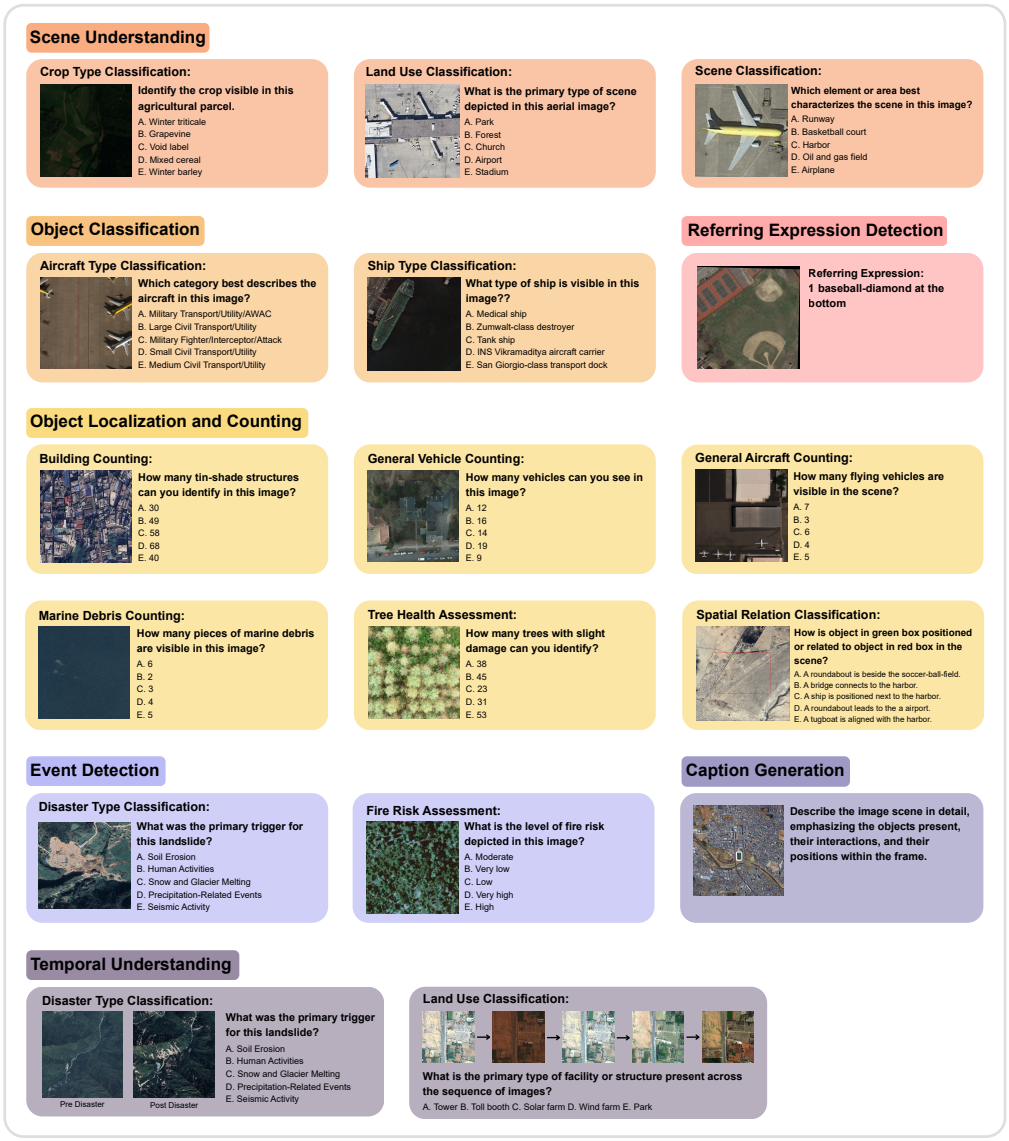


Figure 8: Qualitative examples from the GeoBench-VLM benchmark.

G Additional Qualitative Case Studies

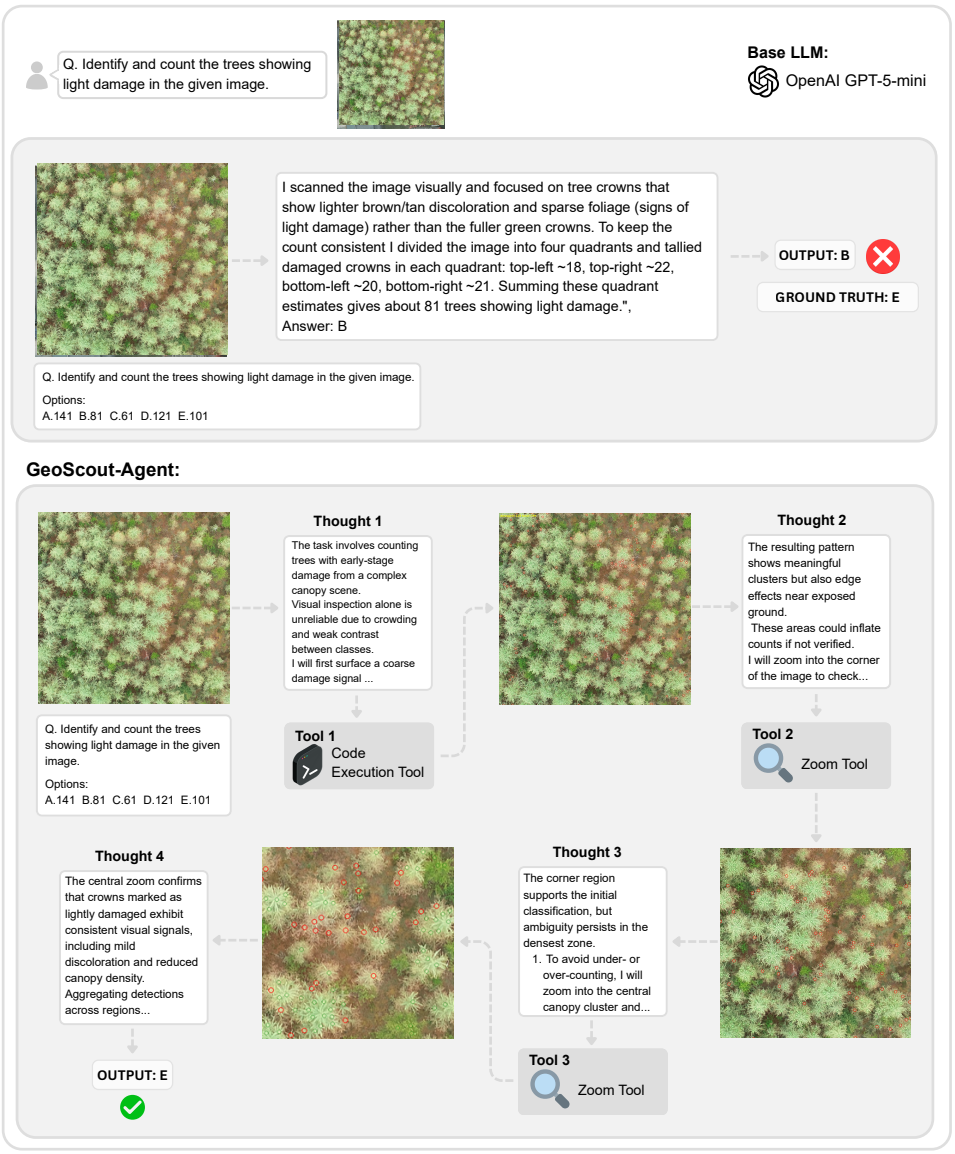


Figure 9: Qualitative case study on tree health assessment. GeoScout-Agent iteratively refines visual evidence through localized inspection, enabling more reliable discrimination of vegetation condition in the presence of clutter, shadows, and resolution variability.

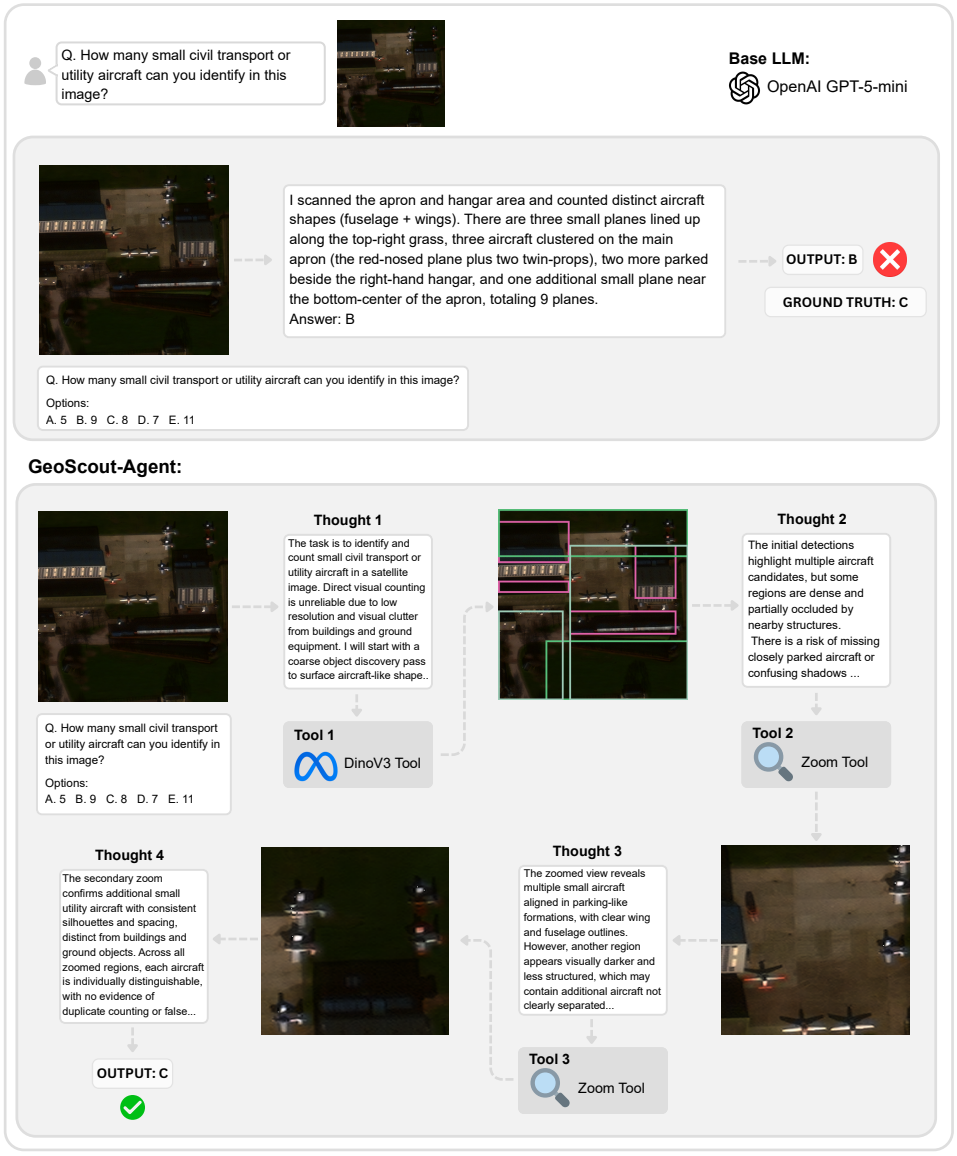


Figure 10: Qualitative case study on specific aircraft type counting. GeoScout-Agent resolves dense layouts and visually similar instances through multi-step object discovery, targeted zooming, and verification, illustrating how tool-assisted reasoning improves fine-grained counting accuracy.

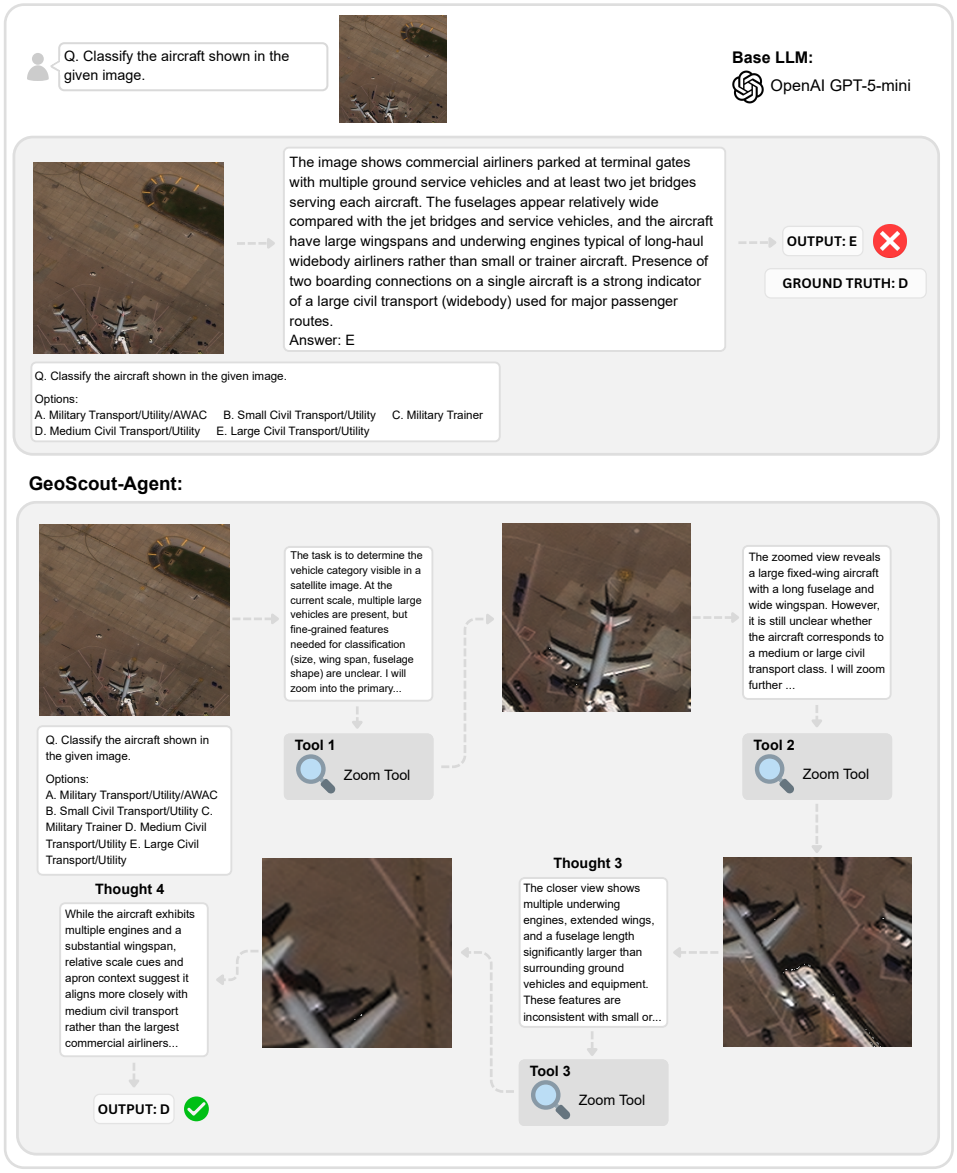


Figure 11: Qualitative case study on aircraft type classification. GeoScout-Agent leverages iterative visual refinement to isolate discriminative structural features, reducing ambiguity between closely related aircraft categories that commonly confound single-pass inference.

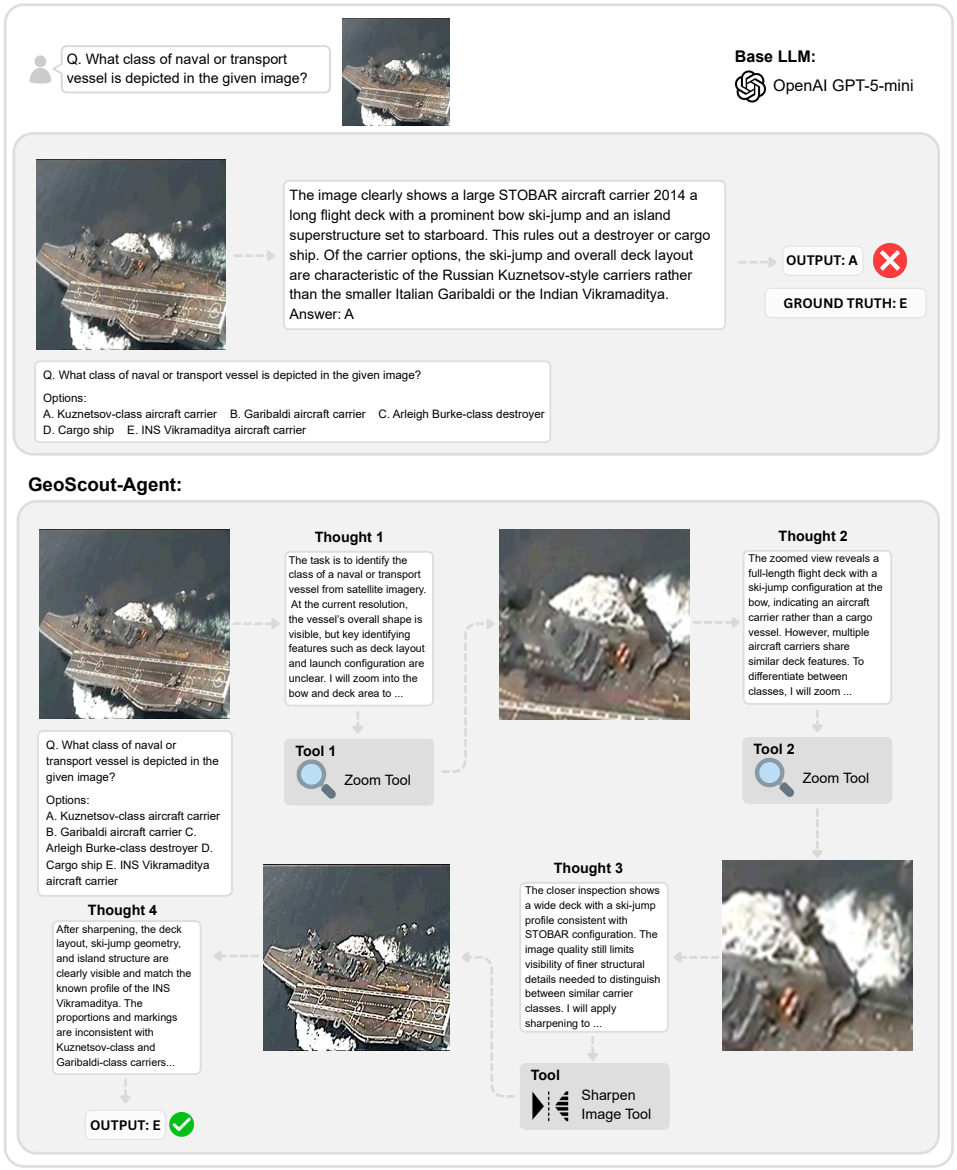


Figure 12: Qualitative case study on Ship Type Classification. GeoScout-Agent applies iterative zooming and sharpening to resolve fine-grained deck and superstructure cues, disambiguating closely related aircraft carrier classes that mislead single-pass LLM inference.

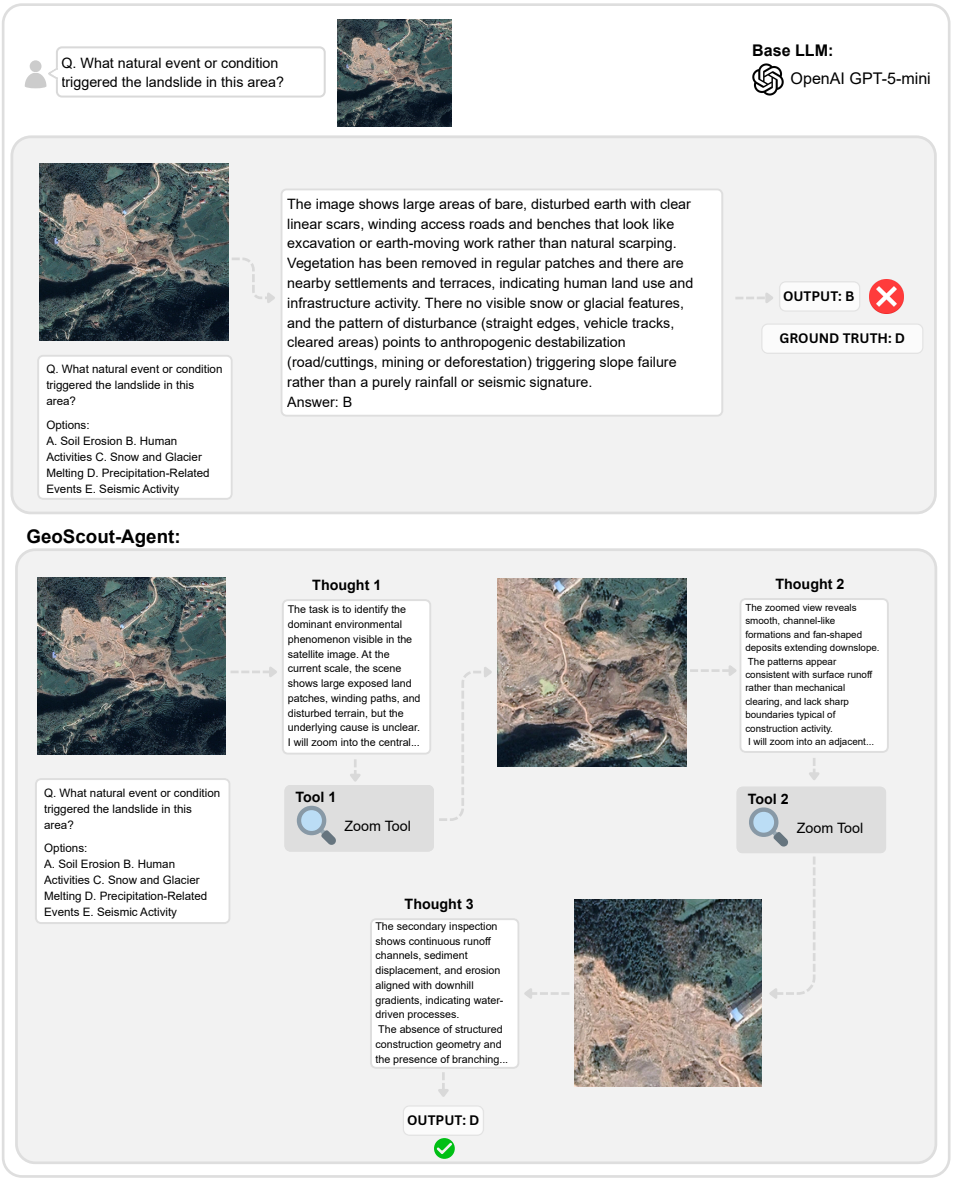
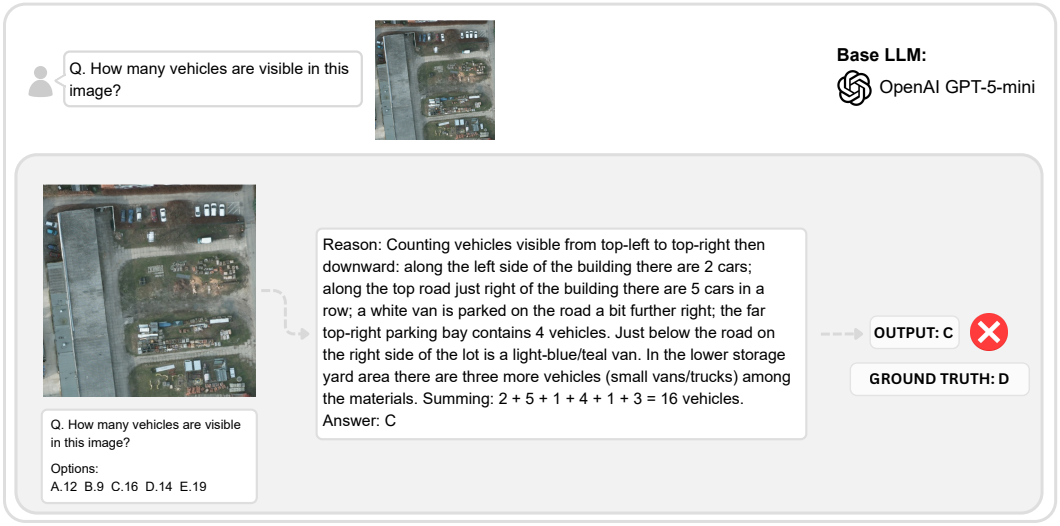


Figure 13: Qualitative case study on Disaster Type Classification. GeoScout-Agent uses iterative zoom-based inspection to distinguish hydrological erosion patterns from anthropogenic disturbance, correctly attributing slope failure to precipitation-driven processes overlooked by single-pass inference.



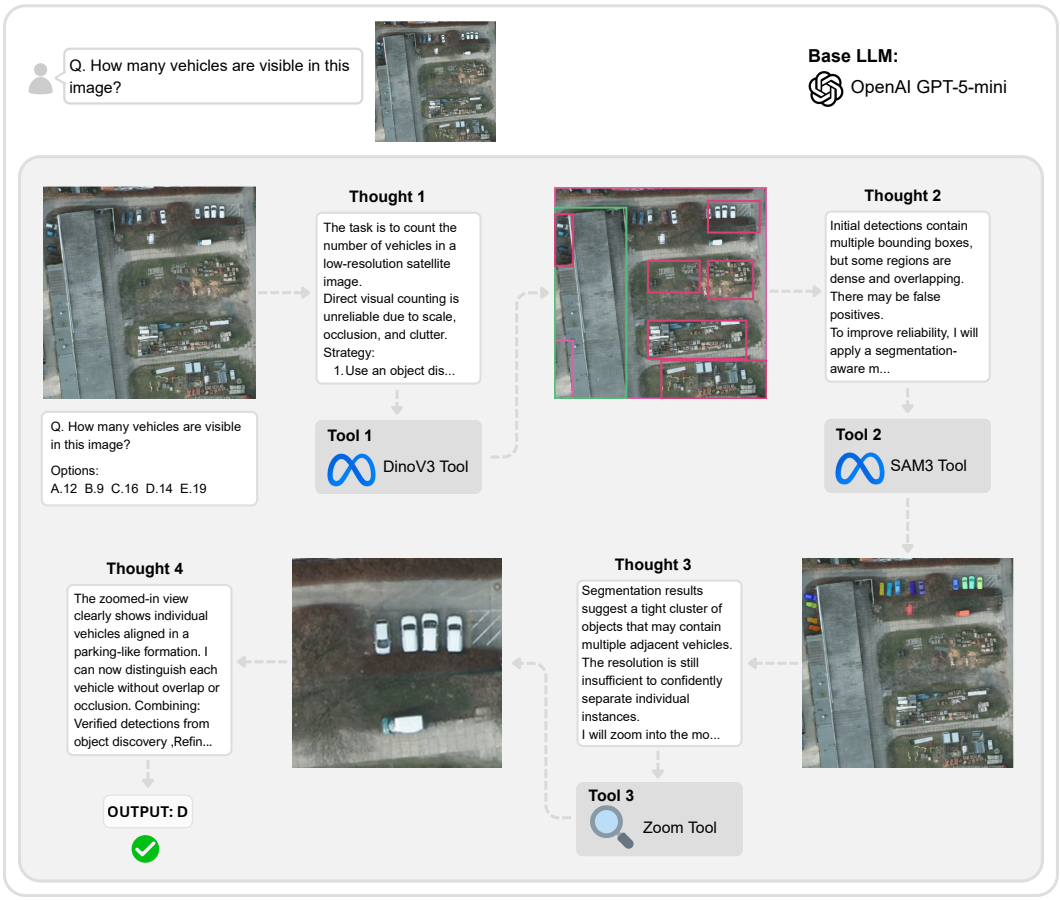


Figure 14: Successful vehicle counting with GeoScout-Agent. The agent incrementally refines its hypothesis through object discovery, segmentation, and targeted zooming, enabling reliable instance separation in a cluttered scene and yielding the correct final prediction via iterative visual verification.