Supplementary: Foundation Models for Remote Sensing: An Analysis of MLLMs for Object Localization.

Darryl Hannan, John Cooper, Dylan White, Tim Doster, Henry Kvinge, and Yijing Watkins
Pacific Northwest National Laboratory
Seattle, WA
darryl.hannan, john.cooper, dylan.white,
timothy.doster, henry.kvinge, yijing.watkins @pnnl.gov

1. Implementation Details

MLLMs We used the HuggingFace implementations for each of the MLLMs [8] and opted to use the instruct variants of each model. For Molmo, we used the standard release of Molmo 7B O and a 4-bit quantized version of Molmo 72B. For Qwen 2.5-VL 7B, we used the public code release and we used the 4-bit quantized version of Qwen 2.5-VL 72B. For Llama 3.2 11B, we used the original release by Meta and we used a 4-bit quantized version of Llama 3.2 90B. All MLLMs were run on a single Nvidia H100 GPU. We set the max new tokens to 800 tokens. We used the default settings for Llama 3.2 and Qwen 2.5-VL and for Molmo we specified a greedy sampling strategy for token generation. The prompts that we used for each model in our zero-shot and fine-tuning experiments are available in Table 1 in the main paper.

Few-shot Details To construct few-shot splits for each of our datasets, we randomly sampled K images for each target category from the dataset (i.e. 3 sets of images for AAP and 1 set of images for xBD/RarePlanes), where K is the number of shots that we are considering. We created 10 seeds for each K-shot split, each consisting of different images and a varying number of annotations. For RarePlanes, we trained our Faster RCNN using Detectron2 [9] for RarePlanes. The Faster RCNN has a ResNet-50 [4] backbone and we used the ResNet 50 weights provided by Detectron2 to initialize our model. We used the standard configuration provided by Detectron2, modifying the images per batch to 2, the base learning rate to 0.0025, and the ROI head batch size to 64. We trained each model on an Nvidia A100 GPU for 1000 iterations, as we find this sufficient for convergence. For AAP and xBD, we used MMDetection [1] to train the Faster RCNNs. We used the standard COCO configuration for MMDetection and the available ResNet 50 pre-trained weights to initialize the model.

Few-shot experiments were conducted using DoRA [6] with rank$= 8$ and $\alpha = 16$. Molmo 7B-D models were trained for 12 epochs at a learning rate of $1e-5$ on each 16-shot task. DoRA weights were merged only for inference, with all inferencing being conducted on the full datasets. Only the latest model checkpoint was saved and used for inferencing. We allowed adaptations to both Molmo's language and vision components. We suspected that false positives in the 0-shot model could be remedied by tailoring model output to a few concise centerpoint annotations. Given Molmo's apparent lack of exposure to overhead imagery, the vision component was adapted on in order to enrich its overhead feature space and allow for slight domain shift. For response generation, top-k, top-p, temperature, and sampling were not used in order to limit the verbosity of the responses and in turn, catastrophic hallucinations.

To produce trainable annotations, we convert the original coco-formatted annotations of the RarePlanes [7] and Aerial Animal Population [3] data sets to VQA-style annotations [5] to be ingested by the model. To construct annotations aligning with Molmo's [2] training strategy, we randomly sample from the template VQA point prompts it provides. That is, each annotation's prompt comprises our label injected into template point prompts seen by Molmo during its original training.

GSD Exploration Details To generate the tiles for our various GSDs across the RarePlanes dataset, we get the original row and column GSDs from the original image metadata and re-scale each image using bi-linear interpolation along both dimensions to match our target GSDs from 0.4 meters per pixel to 1.1 meters per pixel. We tile these re-scaled images with 200 pixels of overlap and only keep tiles where at least one annotation bounding box is at least 50% contained in a tile. If the re-scaled image is smaller than the specified tile size, that image is skipped in our resulting set of tiles for that GSD. Finally, for the set of tiles with variable tile size for fixed spatial content, we take the

initial 1120 pixels by 1120 pixels per tile at 0.4 meters per pixel and re-scale each image to each of the larger GSDs. For each tile-size and GSD, we used Molmo 72B with max new tokens set to 600 tokens, providing each tile and text prompt "Count the number of aircraft in the image." We then use the centerpoints from the centers of the re-scaled bounding boxes to compare to the model's predicted output centerpoints, collapsing all labeled aircraft objects in the RarePlanes annotations into one "aircraft" category.

2. Llama Model Outputs

As mentioned in the main paper, Llama 3.2 is unable to output valid coordinates, despite the prompt explicitly asking it to do so. Here are some sample outputs that it gave us:

1. The image shows a runway and a parking lot, with four airplanes visible.
   The coordinates of the airplanes are: [0.000000, 0.000000, 0.000000, 0.000000] [0.000000, 0.000000, 0.000000, 0.000000] [0.000000, 0.000000, 0.000000, 0.000000] [0.000000, 0.000000, 0.000000, 0.000000]
2. There are 3 airplanes in the image. The coordinates of the airplanes are: [1] [1017, 1221, 1043, 1264] [2] [1022, 1213, 1051, 1256] [3] [1048, 1226, 1057, 1271]
3. $[0, 0, 0, 0.5][0.85, 1, 0.5, 1]$

It does give valid outputs that are able to be parsed. However, the coordinates frequently make no sense in reference to the underlying image. Most of the time, it scales the coordinates from the range 0 to 1. However, even when it does output coordinates in this format, it frequently does not correspond to valid locations in the image. For instance, the third output looks like it might be valid, however, these points do not actually correspond to the planes in the image. The most common output that we observed was similar to the first output, where the model might get the correct number of planes (or might not) and will simply output a sequence of zeros for the coordinates, resulting in a complete failure scenario. As we made adjustments to the prompt to try to circumvent these issues, we found that Llama was difficult to adapt and had trouble following more detailed system prompts. Note that these issues to do preclude Llama from being used for other tasks that do not require precise localization output, such as object counting, scene description, etc. However, our experiments indicate that without fine-tuning, Llama is not a reliable localization model.

3. Additional Figures

The remainder of the figures in the supplementary illustrate specific success and failure scenarios for various models across each of the datasets.

References

[1] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 1

[2] Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, et al. Molmo and pixmo: Open weights and open data for state-of-the-art multimodal models. *arXiv preprint arXiv:2409.17146*, 2024. 1

[3] Jasper AJ Eikelboom, Johan Wind, Eline van de Ven, Lekishon M Kenana, Bradley Schroder, Henrik J de Knegt, Frank van Langevelde, and Herbert HT Prins. Improving the precision and accuracy of animal population estimates with aerial image object detection. *Methods in Ecology and Evolution*, 10(11):1875–1887, 2019. 1

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[5] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. 2023. 1

[6] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*, 2024. 1

[7] Jacob Shermeyer, Thomas Hossler, Adam Van Etten, Daniel Hogan, Ryan Lewis, and Daeil Kim. Rareplanes: Synthetic data takes flight. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 207–217, 2021. 1

[8] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, 2020. Association for Computational Linguistics. 1

[9] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2.

https://github.com/facebookresearch/detectron2, 2019. 1

Figure 1. Image from xBD dataset with Molmo 72B labels (ground truth represented by green dots and predictions represented by red Xs). This illustrates the common failure scenario that is discussed in the main text, where models will sometimes generate a sequence of many detections in a line. We are uncertain what results in this behavior but we notice it more with small models.

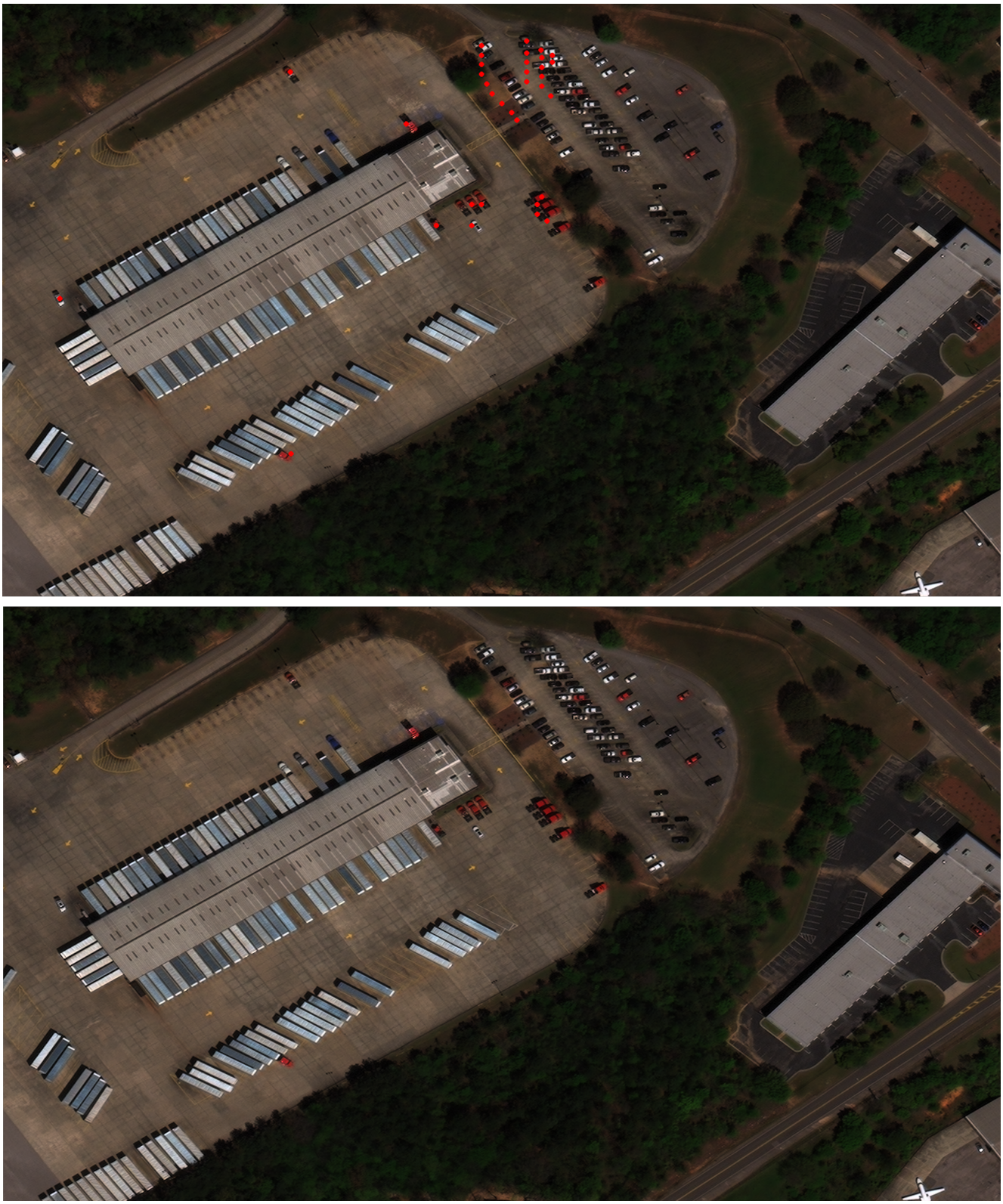

Figure 2. Above: hallucinations resulting from using the prompt "Place a point on each {category} in the image", with a top-p= 0.9, top-k= 50, and temperature= 0.6. Below: reduced hallucinations resulting from using the prompt "Where are the {category}?", and without using top-p, top-k, and temperature. Red dots indicate predictions.

Figure 3. Illustration of an example from RarePlanes using Molmo 72B (ground truth represented by green dots and predictions represented by red Xs). Here, the model successfully detects most aircraft in the scene, despite the terminal providing many distractors. It even detects one of the two aircraft that appear at the edge of the image, suggesting that it is able to detect parts of planes.

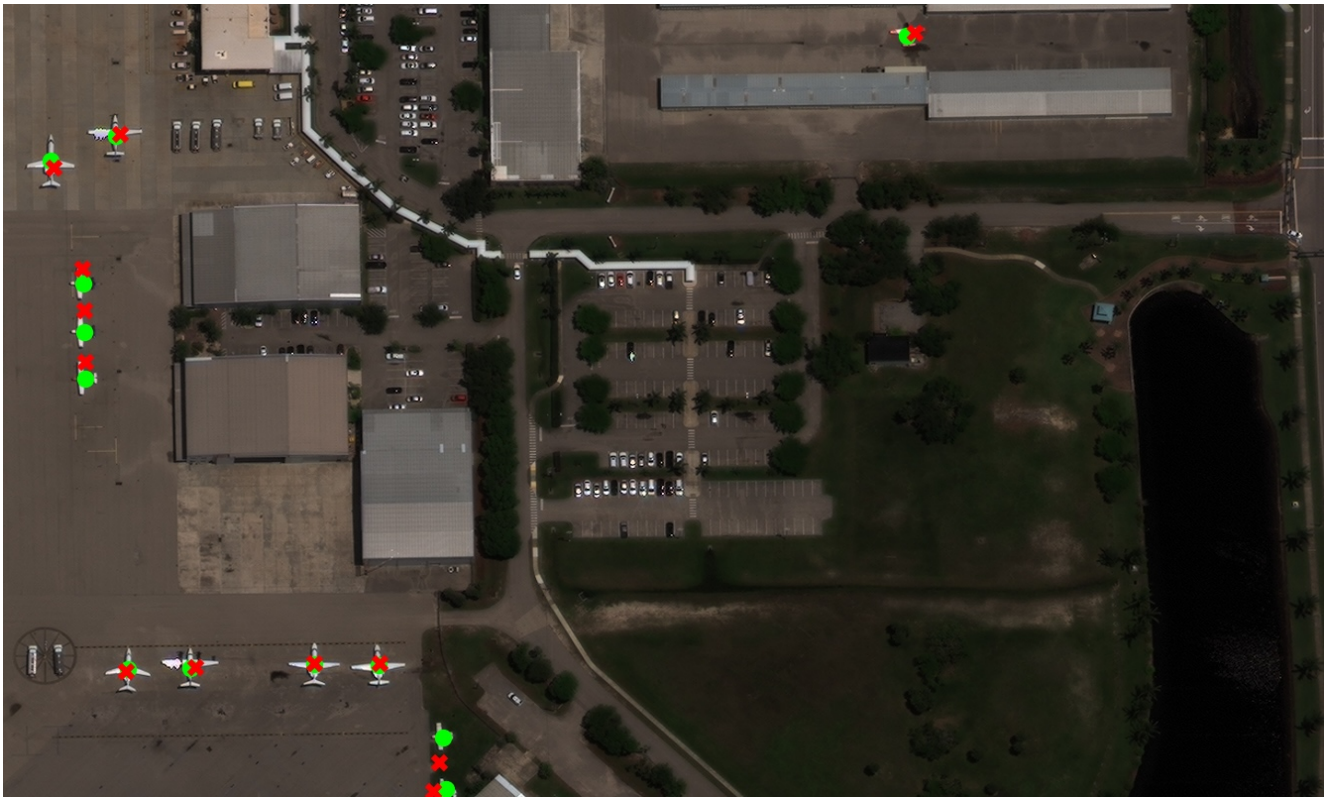

Figure 4. Illustration of an example from RarePlanes using Molmo 72B (ground truth represented by green dots and predictions represented by red Xs). Here, the model successfully detects all aircraft in the image, despite variations in size and orientation.

Figure 5. Illustration of an example from RarePlanes using Molmo 72B (ground truth represented by green dots and predictions represented by red Xs). The model successfully predicts most planes in the image, despite the large number of targets and potential distractors. It misses a plane that is in close quarters to other planes and it misses two planes that are partially obscured.

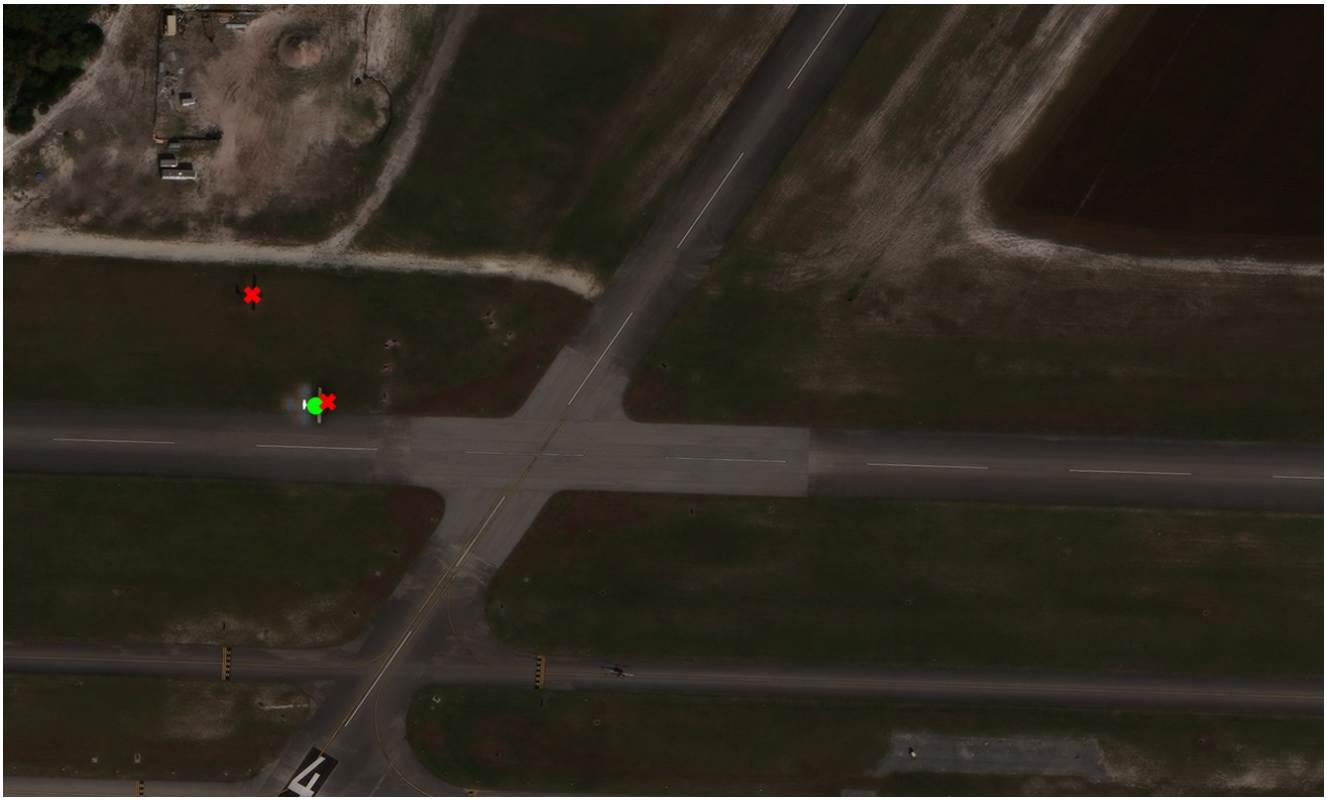

Figure 6. Illustration of a failure case on RarePlanes using Molmo 72B (ground truth represented by green dots and predictions represented by red Xs). An example of a failure where the model detects the plane's shadow as an additional plane. We noticed models making this mistake in other datasets as well.

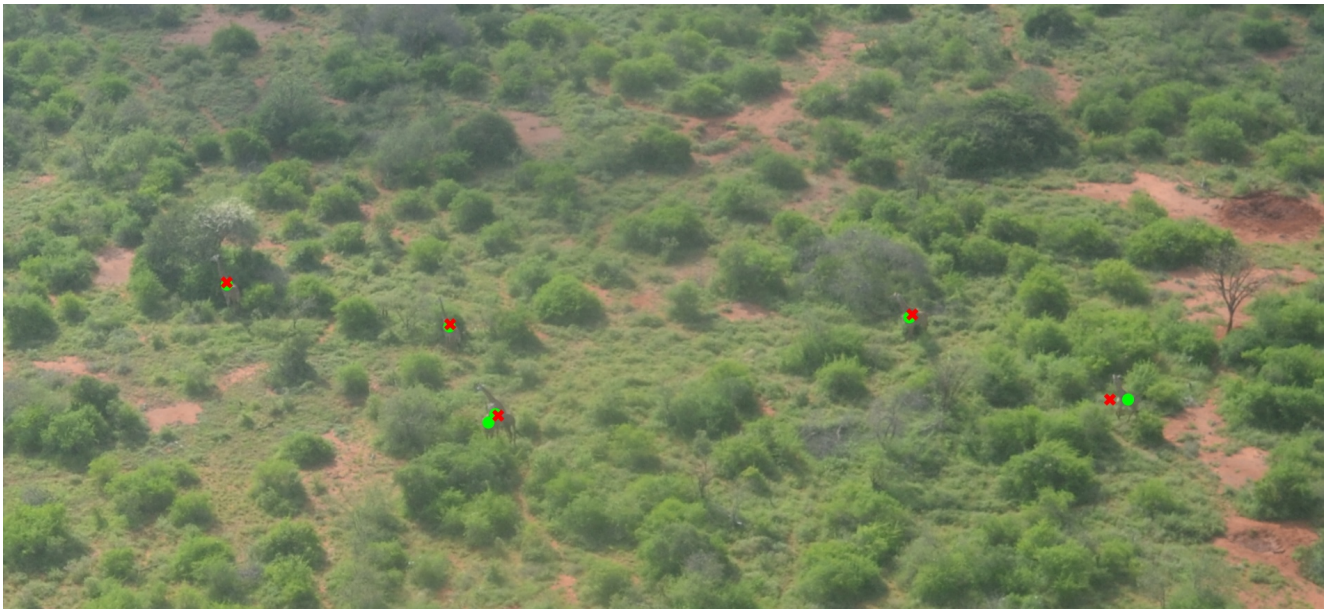

Figure 7. Zoomed in illustration of a success case on the Animal Population dataset using Qwen 2.5-VL 7B (ground truth represented by green dots and predictions represented by red Xs). Note how difficult it is to distinguish the giraffes from the background, nevertheless, Qwen successfully locates each one.

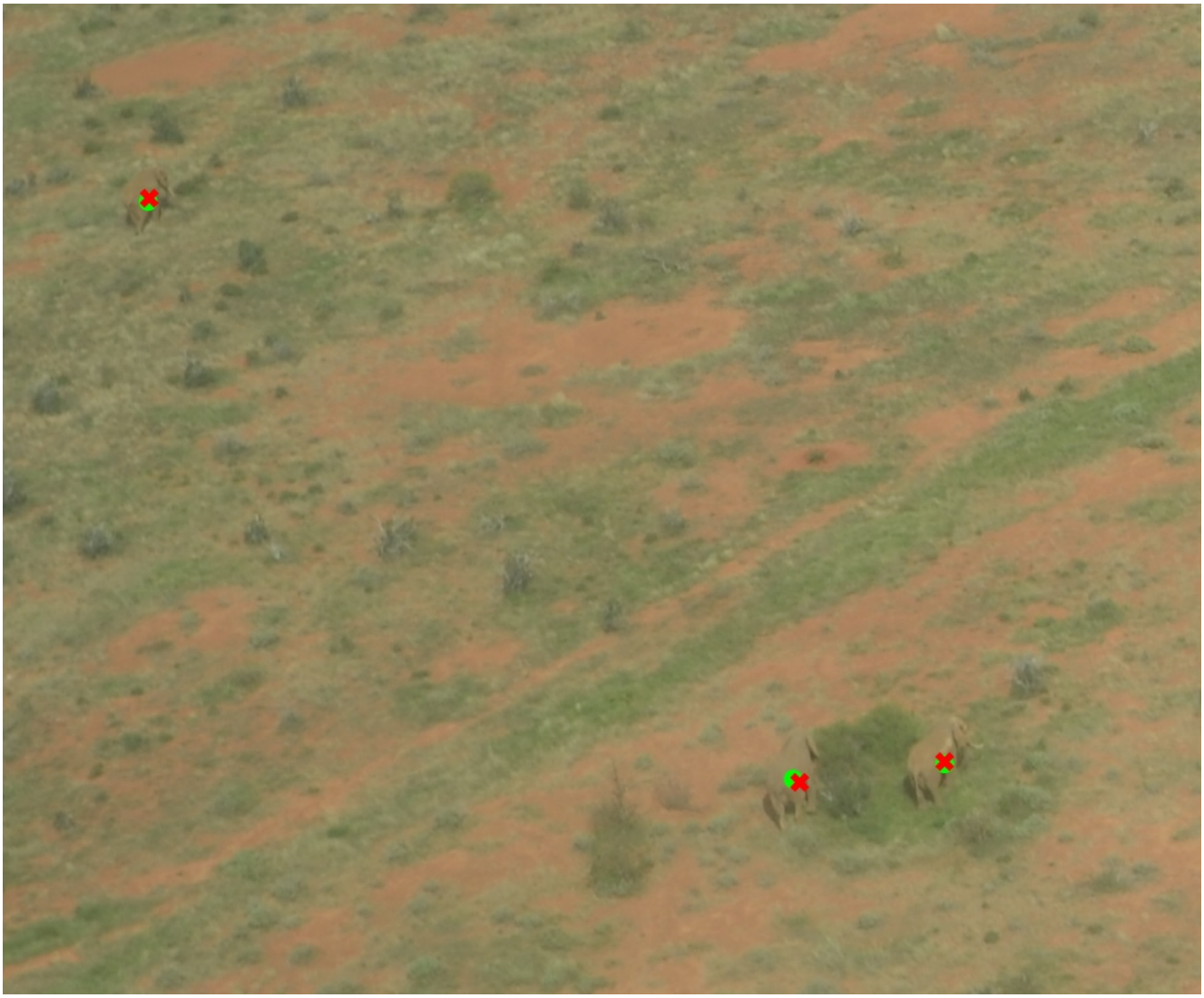

Figure 8. Zoomed in illustration of a success case on the Animal Population dataset using Qwen 2.5-VL 7B (ground truth represented by green dots and predictions represented by red Xs). Qwen successfully locates the elephants in the image. While they initially appear distinct, Figure 9 contains the zoomed out version of the image, where the elephants are no longer clear and are actually quite small relative to the scene.

Figure 9. Full image from the Animal Population dataset with Qwen 2.5-VL 7B labels (ground truth represented by green dots and predictions represented by red Xs).

Figure 10. Zoomed in image from the Animal Population dataset with Qwen 2.5-VL 7B labels (ground truth represented by green dots and predictions represented by red Xs). This illustrates a common failure case where Qwen places a point on a group of animals, rather than individually identifying each one. This substantially hurts the AP of the model.

Figure 11. Zoomed in image from the Animal Population dataset with Qwen 2.5-VL 7B labels (ground truth represented by green dots and predictions represented by red Xs). This is a less severe failure case, more aligned with the types of mistakes one might expect to see, where one individual in a group of animals is missed.

Figure 12. Image from xBD dataset with Molmo 72B labels (ground truth represented by green dots and predictions represented by red Xs). This is an example of a scenario where the model was relatively successful. It definitely possesses the required knowledge to detect buildings from an overhead angle. However, it misses certain buildings, especially when they are quite small, with some of them being fully subsumed by the small dots that we placed on the image.

Figure 13. Image from xBD dataset with Molmo 72B labels (ground truth represented by green dots and predictions represented by red Xs). Another scene in which Molmo was relatively successful. In this case, it again detected many of the homes, which are already relatively small, mostly missing smaller buildings which appear to be sheds or garages.

Figure 14. Image from xBD dataset with Molmo 72B labels (ground truth represented by green dots and predictions represented by red Xs). An exmaple of a failure scenario, where there are too many houses in the image for molmo to detect. It is worth noting that it does get many of the buildings. However, we found that even with expanding the generated token limit, Molmo falls apart past a certain number of objects in a given image, likely due to the distribution of object counts that it was trained on.

Figure 15. Image from xBD dataset with Molmo 72B labels (ground truth represented by green dots and predictions represented by red Xs). Another example of a failure scenario, where in this case the model incorrectly segments a larger building into multiple smaller buildings.