MolmoPoint
Better Pointing for VLMs with Grounding Tokens

Christopher Clark[♥1] **Yue Yang**[♥1] **Jae Sung Park**[♥1] **Zixian Ma**[1,2] **Jieyu Zhang**[1,2] **Rohun Tripathi**[1]
Mohammadreza Salehi[1,2] **Sangho Lee**[1] **Taira Anderson**[1] **Winson Han**[1] **Ranjay Krishna**[♥1,2]

[1]Allen Institute for AI, [2]University of Washington

♥ marks core contributors

🤗 **Models:** MolmoPoint-8B MolmoPoint-GUI-8B MolmoPoint-Vid-8B

Data: MolmoPoint-GUISyn MolmoPoint-TrackAny MolmoPoint-TrackSyn

Code: https://github.com/allenai/molmo2 https://github.com/allenai/MolmoPoint-GUISyn

🤗 **Demos:** MolmoPoint-8B MolmoPoint-GUI-8B

Contact: molmo@allenai.org

Abstract

Ai2

Grounding has become a fundamental capability of vision-language models (VLMs). Most existing VLMs point by generating coordinates as part of their text output, which requires learning a complicated coordinate system and results in a high token count. Instead, we propose a more intuitive pointing mechanism that directly selects the visual tokens that contain the target concept. Our model generates a special pointing token that cross-attends to the input image or video tokens and selects the appropriate one. To make this model more fine-grained, we follow these pointing tokens with an additional special token that selects a fine-grained subpatch within the initially selected region, and then a third token that specifies a location within that subpatch. We further show that performance improves by generating points sequentially in a consistent order, encoding the relative position of the previously selected point, and including a special *no-more-points* class when selecting visual tokens. Using this method, we set a new state-of-the-art on image pointing (70.7% on PointBench), set a new state-of-the-art among fully open models on GUI pointing (61.1% on ScreenSpotPro), and improve video pointing (59.1% human preference win rate vs. a text coordinate baseline) and tracking (+6.3% gain on Molmo2Track). We additionally show that our method achieves much higher sample efficiency and discuss the qualitative differences that emerge from this design change.

1 Introduction

Grounding through pointing is a key capability for vision-language models (VLMs). Pointing has direct applications to robotics, where points have been shown to be an effective way for VLMs to build plans for grasping or navigation [37, 66, 86]. Computer user agents have increasingly used pointing to determine how to interact with graphical user interfaces (GUIs) [69, 100, 58, 74]. Pointing can also be used with chain-of-thought to improve performance on tasks like counting [22, 19], and it can be used to refer back to visual input when communicating with users to provide clearer and more interpretable responses.

VLMs typically point in one of two ways: by directly generating text coordinates [22, 84, 28], or by generating special tokens that correspond to discretized coordinate bins [46, 14]. Instead, as shown in Figure 1, we

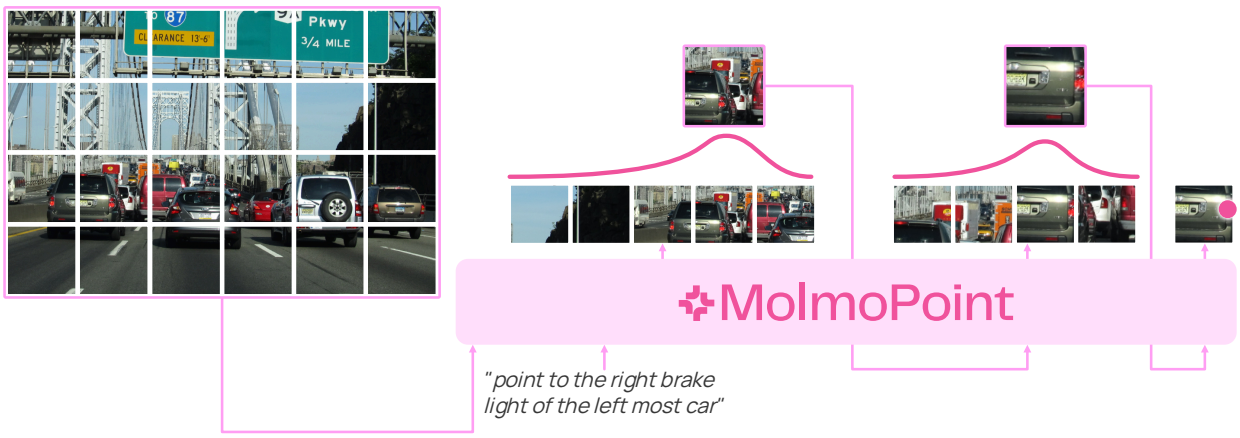


Figure 1 Overview of MolmoPoint. To point, our model scores coarse-grained image patches using the LLM's hidden states, then scores fine-grained subpatches from the highest scoring patch using ViT image features, and then selects a point within the highest scoring subpatch.

propose to use *grounding tokens* that directly select visual tokens from the input video or image.

To predict a point, the model emits three special grounding tokens, `<PATCH>`, `<SUBPATCH>`, and `<LOCATION>`, that generate a point in a coarse-to-fine manner. The `<PATCH>` token is used to select a coarse-grained patch in the input image (or video) by attending to the hidden states of the LLM's visual tokens. The `<SUBPATCH>` token selects a subpatch within that patch by attending to the ViT features of finer-grained patches within that patch. Finally, the `<LOCATION>` token selects a point within the subpatch. When used as input, the `<PATCH>` token and `<SUBPATCH>` token use embeddings derived from the selected patch and subpatch. This allows the model to carry forward location information as it generates future tokens.

To give the model additional awareness of what it has already pointed to, we apply rotary embeddings (RoPE) [64] when selecting a patch to encode how far candidate patches are from the patch selected by the *previous* `<PATCH>` token. This encoding makes it easier for the model to generate consistent, ordered points and avoid double-pointing. We also allow `<PATCH>` tokens to emit a no-more-points class instead of selecting a token to indicate that the model should stop pointing. We show that this prevents degenerate behavior where the model generates an excessive number of points.

Our approach has several practical advantages. First, the model no longer needs to learn or memorize a coordinate system, which we show makes learning faster and improves generalization across image resolutions unseen during training. Second, it reduces the number of output tokens required to represent each point, lowering the decoding cost and improving inference latency. Third, it more tightly couples visual recognition and pointing: if the model has already encoded an object, action, or part in the hidden state of a visual token, it becomes trivial to point to that content by generating a query vector that matches its embedding. We show that this leads to stronger pointing performance and shows signs of improving transfer to tasks beyond grounding.

To explore this approach, we train three models: (1) **MolmoPoint-8B**: a general-purpose image and video VLM following the Molmo2 pipeline, (2) **MolmoPoint-GUI-8B**: a model specialized for GUI pointing, and (3) **MolmoPoint-Vid-8B**: a lighter-weight model specialized for video pointing. To train MolmoPoint-GUI-8B, we construct **MolmoPoint-GUISyn**, a new synthetic dataset of high-resolution GUI grounding examples by extending the code-guided data generation method of CoSyn [90]. To improve tracking in MolmoPoint-8B, we also contribute **MolmoPoint-Track**, a dataset of human-annotated and synthetic tracks for broader object and scene coverage.

We evaluate these models across many pointing tasks. For natural images, MolmoPoint-8B sets a new SoTA on PointBench [16] and PixMo-Points [22], beating the previous methods by 2 points and 4 points, respectively. For GUI pointing, we find MolmoPoint-GUI-8B achieves over 5 points better on ScreenSpotPro [39] and 4 points better on OSWorldG [81] compared to a baseline using text coordinates, and is SoTA among models of

a similar size that have open data. For video pointing, MolmoPoint-8B shows a several-point gain on counting metrics and better human preference scores compared to Molmo2, despite being trained on the same data, and MolmoPoint-Vid-8B further improves these metrics. For video tracking, MolmoPoint-8B reaches 62.5 on $\mathcal{J}\&\mathcal{F}$ vs 56.7 for Molmo2 and shows large gains from both our new data and model design. We also show that our approach improves training and sample efficiency and has notable qualitative effects on the pointing behavior. We will release our models, code, and data.

2 Related Work

Generating Coordinates. Generating text coordinates or discrete tokens for grounding is an old approach for VLMs [71, 14, 46, 45]. Large-scale pointing datasets such as PixMo-Points [22] have allowed VLMs to handle pointing across a wide range of objects and images [22], and many recent VLMs have adopted this capability [28, 84, 44, 93, 72, 8]. MolmoPoint-8B shows that using grounding tokens can provide a stronger and more efficient way to learn this skill.

GUI Grounding. Many recent works have developed models that use pointing to interact with graphical user interfaces [74, 76, 42]. Existing methods often try to improve performance by enhancing data generation [58, 80, 24, 15] or by using reinforcement learning [69, 100, 94, 67]. Other works have improved GUI grounding through agentic, multi-step strategies such as zooming in and cropping the input screenshot [100, 96], although this comes at the expense of higher compute costs. Our work shows that improving the point representation can also significantly enhance GUI grounding.

GUI Grounding Datasets. Existing GUI grounding datasets have been built both purely synthetically [90, 88, 29, 77, 29] and with humans [32, 11, 24]. Our MolmoPoint-GUISyn differs in that it focuses on high-resolution images and greater diversity across operating systems, websites, software, apps, resolutions, and aspect ratios. MolmoPoint-GUISyn also provides extremely dense annotations (54 points per image on average), making it very efficient to train on using *message-trees* to group all annotations for an image into a single training sequence [19].

Video Grounding. Open-vocabulary video grounding is still generally done by specialized models [82, 7, 41, 2], with only a few VLMs supporting this capability [19, 28]. We believe that grounding should not be limited to images, which is partly why we build on top of the Molmo2 models that support video pointing. Our results suggest that token referencing can help in this domain as well.

Grounding Tokens. Grounding tokens have been used for tasks such as image segmentation [8, 35, 7, 59] or depth estimation [36, 9]. These methods typically employ a pre-trained decoder that constructs the grounded output from tokens. In contrast, our method decodes grounding tokens through lightweight projectors on top of the hidden states, removing the need for pre-trained decoders.

More similar to our work, PaDT [65] adds tokens to the model's vocabulary using hidden states of input vision tokens, which allows generated tokens to similarly cross-attend to the input visual tokens. However, their approach uses a separate decoder to obtain bounding boxes or other grounding information from those tokens, whereas our method uses the spatial location of visual tokens, along with refinement with additional tokens, to point. Our method also applies this approach to videos and GUIs.

GUI-Actor [76] also allows cross-attention between a special `<ACTOR>` token and visual patches; however, it does not add refinement stages to allow high-precision pointing and only applies their methods to GUIs and single points.

3 Method

Our approach trains the model to point by directly selecting which visual token contains the target object and then refining that location by generating additional tokens. We describe it in more detail below.

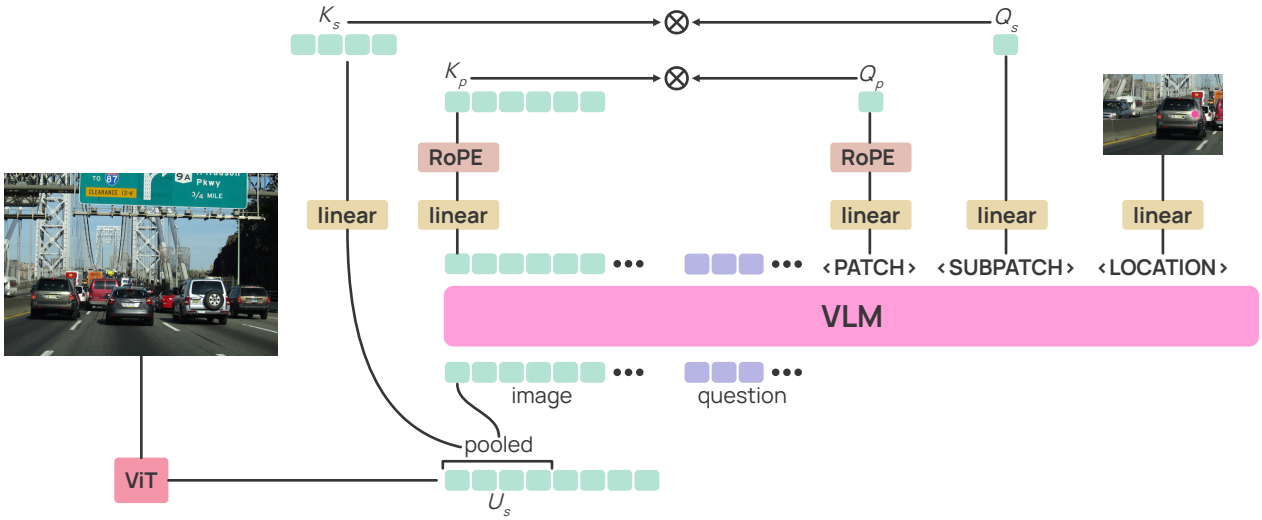


Figure 2 Pointing with grounding tokens. Keys are built from image tokens and ViT patch features, and queries are built from the <PATCH> token and <SUBPATCH> token hidden states, to score patches and subpatches. The <LOCATION> token predicts the final output points within the highest scoring subpatch.

3.1 Patch Selection

First, we add a special <PATCH> token to the model's vocabulary. When this token is generated, a query vector is built from its hidden state:

$$q_p = W_{pq}\text{Norm}(h_p)$$

Here W_{pq} is a learned parameter with shape $M \times D$, h_p is the hidden state of the <PATCH> token with shape $D \times 1$, Norm is a layer norm layer, D is the model's hidden size, and M is a hyper-parameter. We also generate key vectors for each <IMAGE> token that embeds visual input as:

$$K_p = W_{pk}\text{Norm}(H_i)$$

where W_{pk} is another learned parameter with shape $M \times D$, H_i are the hidden states of the <IMAGE> tokens with shape $D \times I$, and I is the number of image tokens. Finally, we score each image token as

$$s_p = K_p^\top q_p / \sqrt{M}$$

The score vector s_p has shape $I \times 1$. During training, we compute the loss of this selection process as:

$$L_p = \text{cross\_entropy}(\text{softmax}(s_p), t_p)$$

where t_p is the ground truth target token. L_p is added directly to the token-level loss from the LLM before that loss is averaged by the number of tokens.

During inference, we select the highest scoring token $p^* = \text{argmax}(s_p)$. During training, we instead use $p^* = t_p$. Then, when <PATCH> token is an input, we add the input embedding of the <IMAGE> token that was selected to its embedding: $q_p + E_i[:, p^*]$, where E_i are input embeddings of the <IMAGE> tokens. This is important so the model is aware of which token it pointed to.

During training, we sort ground truth points so that the <IMAGE> tokens the <PATCH> tokens select are ordered based on where they appear in the input sequence. We mask out <IMAGE> tokens that come before previously selected <IMAGE> tokens during both training and inference to enforce this pattern.

3.2 Location Refinement

In most VLMs, image tokens are constructed by pooling multiple patches from the underlying ViT. For example, in Molmo2 models, each <IMAGE> token is built from 4 ViT patches that each cover 14x14 pixels, so it represents a 28x28 pixel area. This is too coarse-grained, so we refine that location by adding additional tokens after the <PATCH> token.

After a <PATCH> token, our model also emits a <SUBPATCH> token that selects one of the ViT patches that were pooled to build $p*$. This is done through dot-product scoring as before. The hidden state of the <SUBPATCH> token h_s is projected to create a query vector q_s, and key vectors K_s are built by projecting the ViT features for the subpatches U_s, where U_s is a $T \times K$ matrix, K is the number of subpatches, and T is the dimensionality of the ViT.

We similarly use the ground-truth subpatch location to compute a loss L_s for this component during training and select a subpatch index s^*. When a <SUBPATCH> token is used as input, its embedding is built from the hidden state of the selected ViT patch: $q_s + W_{se} U_s[:, s^*]$ where W_{se} with shape $D \times T$ projects the ViT patch feature to the LLM's dimension. Adding this embedding indicates to the LLM which subpatch was selected and gives the model access to the unpooled features of the selected patch, which we find important when trying to further refine the location.

This gives us a 14x14 resolution, which can still be too coarse-grained. To produce a precise point, we emit a final <LOCATION> token. The hidden state of the <LOCATION> token is used to predict one of 9 locations within the subpatch (arranged in a 3x3 grid) using a single linear layer. With 14x14 ViT patches, this results in a precision of about 4.7 pixels. Unlike pointing with text coordinates, this method maintains a 4-pixel resolution regardless of input size, potentially enabling fine-grained pointing even with ultra-HD images.

3.3 Rotary Embedding

We add rotary embeddings to better encode how <IMAGE> tokens are positioned relative to the previously selected <PATCH> token. This is important to help the model follow the sorted order of points or to track what frames the previous points were generated for when doing video pointing.

This is implemented by rotating the <PATCH> token key and query vectors:

$$s_p = \mathrm{Rot}(K_p, p_i)^\top \mathrm{Rot}(q_p, p_q)/\sqrt{M}$$

Where p_i contains the <IMAGE> token position $[0, 1, 2, ..., I]$ and p_q is the image position selected by the previous <PATCH> token, or 0 if there is no such <PATCH> token.

3.4 No-More-Points Class

One issue with this approach is that if the model chooses to generate a <PATCH> token, it is forced to select a point, even if none of the scores in s_p are high. We observe that this can sometimes lead to degenerate output, where the model generates an excessive number of points.

To solve this, we add a special *no-more-points* class with a fixed key embedding that the <PATCH> token can attend to, meaning we have:

$$K_p = [W_{pk}\mathrm{Norm}(h_i); h_{done}]$$

Where h_{done} is a learned $M \times 1$ vector. We use a position of 0 for h_{done} when applying rotary embeddings. If the no-more-points class is selected, the model is prevented from generating a <SUBPATCH> token and stops pointing.

4 Training and Inference

We train three models using this proposed method. We present high-level details of how they are trained but leave the specifics to the appendix.

Figure 3 Overview of the generation of MolmoPoint-GUISyn. We prompt an LLM to generate the HTML for the screenshot and extract all bounding boxes of its UI elements. Then we use LLMs to annotate each bounding box with its interaction intents.

4.1 Implementation

During pre-processing, we map input points to the corresponding target `<IMAGE>` token index, ViT patch index, and location index, and use those triples as additional input to the model. Our text input for points follows the Molmo2 [19] format, but replaces the string coordinates with the grounding tokens, including an additional `<PATCH>` token at the end of each list of points that is assigned the no-more-points class. This reduces the number of tokens per coordinate from 8 (6 digits and 2 spaces) to 3. For video, we also remove the text timestamps used by Molmo2 since they can be recovered based on which `<IMAGE>` token was selected, further reducing the token count.

As with Molmo2, we also give an integer object ID for each point, but place it after the coordinates instead of before. Following Molmo2, we use a separate learning rate and gradient norm for the new pointing parameters. In general, we set the learning rate to match that used for the image-text connector parameters. We set $M = 512$ for all experiments. In all training runs, we use packing and message-trees to support training on multiple examples per sequence [19].

4.2 Inference

During inference, we cache the keys of the image tokens and ViT patches during prefilling. This adds additional memory overhead, but the low-dimensionality of the keys means this uses roughly the same memory as the cached keys and values for 1-2 LLM layers, and it is only required for the image tokens.

We constrain the model to generate a `<SUBPATCH>` token and `<LOCATION>` token after each `<PATCH>` token, and to only select `<IMAGE>` tokens that are the same, or after, any `<IMAGE>` token it has already selected in the input sequence, so output points are ordered correctly. We also prevent the model from generating multiple points with the same `<IMAGE>` token and ViT subpatch since we observe that this is almost always a case of the model pointing to the same thing twice. If the model selects the no-more-points class, we constrain the model to generate the `">` token, which ends a list of points in the Molmo2 pointing format.

To convert the selected patches back into coordinates, we retain a map of token_id → coordinates for every ViT patch during pre-processing and combine it with the location predictions to get the output point.

4.3 Models

MolmoPoint-8B. We conduct a full end-to-end training run following the pipeline of Molmo2-8B. We use a larger batch size of 160 to better utilize the hardware we have available and lower the number of training steps from 30000 to 22,000 to compensate. To improve tracking, we also incorporate MolmoPoint-Track, a new dataset of human-annotated and synthetic tracks (see below). We also slightly adjust the training mixture to better exploit the improved learning efficiency of the pointing data (see the appendix for details).

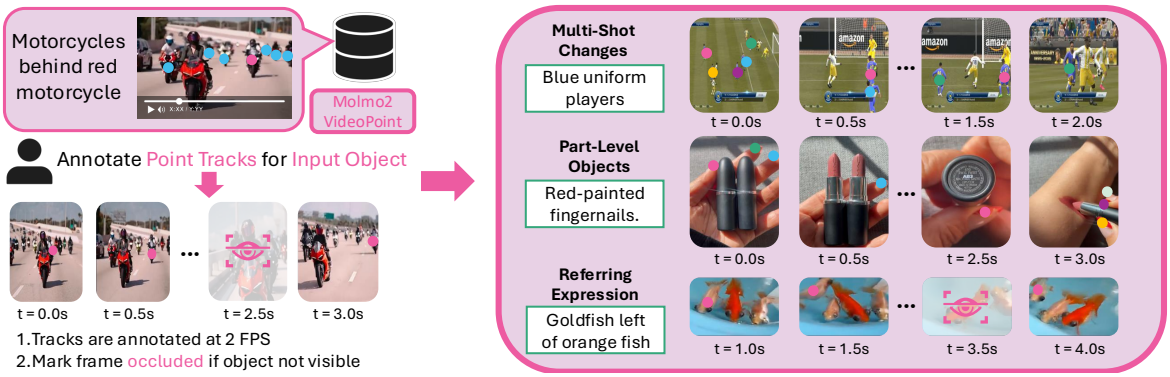


Figure 4 MolmoPoint-TrackAny: human-annotated point-to-track extension. Annotators are given a text query and an object of interest, and provide point tracks while marking frames as occluded when the object is not visible.

MolmoPoint-GUI-8B. The image pointing data in the Molmo2 mixture does not contain many instructional/GUI examples. To train a model better optimized for this task, we build MolmoPoint-GUISyn, a code-guided synthetic GUI instructional dataset (see below for details), and fine-tune on it for 2000 steps with a batch size of 128 while increasing the image resolution to 48 crops per image.

MolmoPoint-Vid-8B. As with Molmo2, we observe that MolmoPoint-8B underperforms the specialized models on video grounding. We therefore also train a specialized video grounding model by finetuning MolmoPoint-8B after the pre-training stage on just video-pointing data for 6000 steps with a batch size of 64 and a max of 128 frames. We then fine-tune it for another 800 steps with a max of 384 frames to support longer videos.

4.4 MolmoPoint-GUISyn

As shown in Figure 3, we extend the code-guided synthetic data generation framework (CoSyn) [90] to screenshot generation, in which we prompt the language model to generate HTML code that mimics digital environments for web, desktop, and mobile. Given access to the underlying HTML code in each screenshot, we use the Playwright library with custom JavaScript to automatically extract bounding boxes for all elements in the screenshot. We then feed the bounding box information to the language model to generate 5 pointing instructions per element that a user may ask when interacting with it. In total, we synthesize 36K screenshots, with 2M densely annotated points and over 10M pointing instructions. Qualitative examples of this data are provided in Figure 8 in the appendix.

4.5 MolmoPoint-Track

Existing tracking datasets with referring expressions, such as Molmo2-VideoTrack [19], were collected by expanding tracks for a fixed set of objects, resulting in limited scene and object diversity. Here, we contribute MolmoPoint-Track, consisting of (1) **MolmoPoint-TrackAny**, human-annotated tracks on videos with any objects and (2) **MolmoPoint-TrackSyn**, synthetic tracks with diverse motion and occlusion patterns. For MolmoPoint-TrackAny, we extend Molmo2-VideoPoint annotations into full tracks via human annotation (Figure 4). For MolmoPoint-TrackSyn, we generate multi-object tracking videos in Blender with complex occlusion and motion dynamics, paired with automatically generated referring queries (Figure 7). See Appendix 11 for collection details and qualitative examples.

5 Results

5.1 Image Pointing

We show results on natural image pointing in Table 1 and Table 2. MolmoPoint-8B is state-of-the-art on PointBench [16], surpassing Molmo2 by almost 2 points, including a 5 point gain in reasoning and spatial reasoning. On PixMo-Points [22] MolmoPoint-8B surpasses Molmo2 by 4 points. Molmo2 and MolmoPoint-8B

Table 1 Point-Bench results. Baseline scores taken from the Point-Bench leaderboard. Qwen3-VL-235B-A22B-Instruct and VisionReasoner-7B scores were taken from Poivre [87], which did not include sub-category scores.

Model	Aff.	Spat.	Reason	Steer.	Count.	Avg
Human	92.3	83.6	87.8	86.3	95.6	89.1
API call only						
Gemini-Robotics-ER-1.5 [1]	69.7	69.7	60.1	**67.5**	68.5	67.1
Gemini-2.5-Pro [20]	72.7	70.3	71.0	41.0	59.2	62.8
Open weights						
Poivre-7B [87]	-	-	-	-	-	67.5
Qwen2.5-VL-32B-Instuct [83]	76.8	60.0	54.4	46.5	57.1	59.0
Qwen2.5-VL-72B-Instuct [83]	76.8	60.0	54.4	46.5	57.1	59.0
Qwen3VL [84]	81.3	65.6	60.6	23.5	61.2	58.5
Qwen3-VL-235B [84]	-	-	-	-	-	58.3
Fully open						
VisionReasoner-7B [44]	-	-	-	-	-	64.7
Molmo-7B-D [22]	82.8	67.7	70.5	28.5	58.7	61.6
Molmo-72B [22]	87.9	70.3	69.4	37.0	54.6	63.8
Molmo-7B-O [22]	84.9	63.1	63.2	45.5	59.7	63.3
Molmo2-4B[19]	82.3	71.8	72.0	41.0	71.4	67.7
Molmo2-8B[19]	84.8	71.3	71.5	44.5	71.4	68.7
Molmo2-O-7B[19]	81.8	69.7	69.4	39.0	72.4	66.5
MolmoPoint						
MolmoPoint-8B	**85.9**	**76.9**	**77.2**	39.0	**74.5**	**70.7**

used the same data and training procedure, so these results show that using grounding tokens significantly boosts pointing capabilities on natural images.

5.2 GUI Pointing

We show results on ScreenSpot-V2 [39], ScreenSpot-Pro [39], and OSWorldG [81]. In addition to other models, we also compare to a baseline, Molmo2-GUI-8B, built by fine-tuning Molmo2-8B on the same data MolmoPoint-GUI-8B was trained on. The Molmo2 data mixture does not contain instruction-point pairs, so MolmoPoint-8B sometimes does not point when given them as input. To fix this, we use constrained decoding for both the Molmo2 models and MolmoPoint-8B (but not MolmoPoint-GUI-8B) to force the model to generate exactly one point. We also show results with *test-time-scaling* where we increase the number of crops during test time to 64. We find that test-time scaling breaks models that use text coordinates, dropping performance to $< 10\%$, presumably due to the model not knowing how to map the larger number of patches to text coordinates. Therefore, we do not use it for other models.

Results are shown in Table 3. Compared to Molmo2, MolmoPoint-8B is even on ScreenSpot-V2 but shows significant improvements on ScreenSpotPro and OSWorldG, again showing the benefit of our pointing method. Fine-tuning with instruction-image data makes MolmoPoint-GUI-8B SoTA among fully open models on all tasks. Open-weight models UI-Venus and MAI-UI show better results, likely because both models utilize large-scale proprietary data collection efforts as well as more elaborate training pipelines that include RL.

We observe a gap of 2 to 9 points between MolmoPoint-GUI-8B and the baseline that uses text coordinates on ScreenSpotPro, showing that our model design is critical for this high performance. We hypothesize that the large gap of 9 points in ScreenSpotPro might be due to grounding tokens having a particularly high impact when dealing with high-resolution input.

Table 2 PixMo-Points results. MolmoPoint-8B surpasses even proprietary models. We collect results for GPT-5.2, Gemini-3, and Qwen3-VL ourselves.

Metric	API-only		Open-weights		Fully-open						Ours
	GPT-5.2	Gemini3-Pro	Qwen3-VL-8B	Qwen3-VL-4B	Molmo-7B-D	Molmo-72B	Molmo-7B-O	Molmo2-4B	Molmo2-8B	Molmo2-O-7B	MolmoPoint-8B
Recall	31.0	77.3	54.3	45.1	76.4	74.9	74.4	83.3	85.5	83.1	**90.4**
Precision	32.9	81.3	53.5	44.2	76.2	74.9	74.6	85.1	86.4	83.7	**89.3**
F1	31.6	77.8	53.4	43.7	75.7	74.5	74.0	83.4	85.2	82.7	**89.2**

Table 3 GUI grounding results. MolmoPoint-GUI-8B is our GUI specialized model finetuned on MolmoPoint-GUISyn (Synthetic GUI dataset) we constructed. *Molmo2-GUI-8B* also fine-tunes on MolmoPoint-GUISyn but without our token referencing mechanism. (64crops) denotes our test-time scaling for inference with more image crops. The best performance of *fully open* models is **bold**. Scores with * are from evaluations in [69, 58].

Model	ScreenSpot-V2	ScreenSpot-Pro	OSWorldG
API call only			
Claude 3.7 [3]	87.6	27.7	-
OpenAI CUA [55]	87.9	23.4	-
Gemini-3-Pro [28]	93.7	72.7	35.5
Open weights			
Holo2-8B [21]	93.2	58.9	70.1
UI-TARS 1.5-7B [58]	94.2	61.6	64.2*
UI-Venus-1.5-8B [69]	95.9	68.4	69.7
Qwen3-VL-8B [6]	92.1*	52.7*	57.5*
MAI-UI-8B [100]	95.2	65.8	60.1
Fully open			
GUI-Actor [76]	90.9	41.8	-
JEDI-7B [80]	91.7	50.2	54.1
GroundCUA-7B [24]	89.3	50.2	67.2
OpenCUA-7B [74]	92.3	50.0	55.3
GTA1-7B [89]	92.4	50.1	60.1
Molmo2-8B[19]	89.5	30.4	54.1
Molmo2-GUI-8B	88.8	52.3	66.1
MolmoPoint			
MolmoPoint-8B	89.8	36.4	54.9
MolmoPoint-8B (64crops)	89.8	39.4	56.5
MolmoPoint-GUI-8B	93.4	60.2	**70.0**
MolmoPoint-GUI-8B (64crops)	**93.9**	**61.1**	**70.0**

5.3 Video Pointing

We evaluate video pointing on BURST-VideoCount [5], Molmo2-VideoCount, and Molmo2-VideoPoint [19]. For the counting datasets, we report accuracy as well as close accuracy, which measures if the number of points is almost correct (computed as $|pred - gt| \leq \Delta$, where $\Delta = 1 + \lfloor 0.05 \times gt \rfloor$). For Molmo2-VideoPoint, we report F1, recall, and precision metrics when matching points to ground-truth segmentation masks. Baseline numbers come from Molmo2 [19].

For MolmoPoint-8B, we see an improvement on both Burst-VC and Molmo2-VC compared to Molmo2-8B,

Table 4 Video counting and pointing results. MolmoPoint-8B scores highest on BURST-VC and MolmoPoint-8B-VP and second highest on MolmoPoint-8B-VC's close accuracy, slightly behind Gemini 2.5 Pro. Best open model results are **bold**.

	BURST VC (test) [5]		Molmo2-VC		Molmo2-VP		
Model	Acc.	Close acc.	Acc.	Close acc.	F1	Recall	Precision
API call only							
GPT-5 [56]	43.1	73.7	35.8	50.3	4.1	4.4	4.2
GPT-5 mini [56]	46.0	73.0	29.8	49.3	2.2	2.2	2.2
Gemini 3 Pro [28]	44.0	71.7	37.1	53.1	20.0	27.4	19.8
Gemini 2.5 Pro [20]	41.6	70.0	35.8	56.5	13.0	14.5	13.6
Gemini 2.5 Flash [20]	38.7	70.0	31.9	48.2	11.1	11.2	12.2
Claude Sonnet 4.5 [4]	42.4	72.6	27.2	45.1	3.5	3.7	4.3
Open weights							
Qwen3-VL-4B [6]	38.9	74.7	25.3	44.3	0.0	0.0	0.0
Qwen3-VL-8B [6]	42.0	74.4	29.6	47.7	1.5	1.5	1.5
Fully open							
Molmo2-4B	61.5	76.1	34.3	56.1	**39.9**	**42.7**	**39.4**
Molmo2-8B	60.8	75.0	35.5	53.3	38.4	39.3	38.7
Molmo2-O-7B	61.6	76.0	33.2	50.5	35.8	35.8	37.9
MolmoPoint							
MolmoPoint-8B	61.6	**76.9**	35.6	54.6	36.2	35.7	37.8
MolmoPoint-Vid-8B	**62.0**	76.3	**36.0**	**58.7**	38.8	39.8	38.8

although we also see a drop in Molmo2-VP. To get a more definitive result, we conduct a human preference evaluation using predictions from both models on 470 video pointing queries (See appendix for details). We find 152 ties, 130 wins for Molmo2, and 188 wins for MolmoPoint-8B. Excluding ties, MolmoPoint-8B has a 59.1% win rate, showing humans prefer MolmoPoint-8B's output. MolmoPoint-Vid-8B sees a more consistent gain, including a full 5 point gain on Molmo2-VC close, surpassing Gemini 3 Pro.

5.4 Tracking Results

We evaluate MolmoPoint-8B on the tracking benchmarks introduced in Molmo2 [19]. Table 5 presents results on academic benchmarks and Table 6 reports results on MolmoPoint-8B-Track across video domains. Following [2, 19], Jaccard and F-measure ($\mathcal{J}\&\mathcal{F}$), which measures segmentation quality, is computed by passing MolmoPoint-8B points as input to SAM2 [60] to obtain segmentation masks. F1 and HOTA [48] evaluate point accuracy directly, where F1 measures whether predicted points fall within the ground-truth segmentation and HOTA further accounts for association consistency across frames.

Overall, MolmoPoint-8B shows substantial gains over Molmo2-8B across tracking benchmarks. Notable improvements come from the multi-object tracking dataset as MolmoPoint-8B reaches **63.5 vs. 62.3** and **72.2 vs. 70.8** $\mathcal{J}\&\mathcal{F}$ on MeViS valid and valid-u splits [23]. On Molmo2-Track, the gains are consistent across all video domains, with overall improvements of **+5.7** $\mathcal{J}\&\mathcal{F}$, **+3.1 F1** and **+2.5 HOTA**. We suspect our grounding tokens enable better grounding and instance-level identification in tracking as well. One exception is ReasonVOS [7] which targets queries that require semantic reasoning rather than precise spatial grounding, thus grounding tokens provide fewer benefits here.

Tracking Ablations

To disentangle our modeling and data contributions, we conduct ablations that sequentially remove grounding tokens and MolmoPoint-Track. Due to computation constraints, all ablation models are trained for 5K steps on heavily upsampled tracking data. As shown in Table 7, both contribute meaningfully to tracking quality.

Table 5 Tracking Results on Academic Benchmark. $\mathcal{J}\&\mathcal{F}$ measures the segmentation mask quality of object tracks. F1 measures whether points fall in the mask, and HOTA [48] further accounts for consistent ID associations.

	MeViS [23]	MeViS [23]			Ref-YT-VOS [61]			Ref-Davis [34]			ReasonVOS [7]		
	valid	valid-u			valid			valid			test		
Model	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}\&\mathcal{F}$	F1	HOTA	$\mathcal{J}\&\mathcal{F}$	F1	HOTA	$\mathcal{J}\&\mathcal{F}$	F1	HOTA	$\mathcal{J}\&\mathcal{F}$	F1	HOTA
API call only													
GPT-5 [56]	23.4	26.5	17.3	14.0	30.9	21.0	18.4	25.2	17.0	11.6	24.7	13.6	10.7
GPT-5 mini [56]	15.7	15.4	8.5	6.8	16.2	7.4	6.2	8.4	3.4	2.3	14.6	4.2	3.4
Gemini 3 Pro [28]	42.5	51.1	42.3	36.0	55.0	49.1	45.5	66.6	60.8	55.7	52.6	48.5	42.1
Gemini 2.5 Pro [20]	40.7	52.8	41.2	35.0	45.1	44.5	40.5	45.6	62.7	56.6	44.0	50.2	42.4
Gemini 2.5 Flash [20]	27.6	31.8	24.0	19.9	36.0	32.8	30.0	31.6	36.7	30.0	26.5	25.8	21.0
Open weights only													
Qwen3-VL-4B [84]	29.7	30.6	23.3	18.7	32.1	29.0	26.5	44.4	33.1	26.9	26.5	17.0	13.5
Qwen3-VL-8B [84]	35.1	34.4	30.1	23.8	48.3	42.1	37.6	41.0	41.6	33.2	24.9	22.3	17.5
Specialized open models													
VideoLISA [7]	44.4	53.2	–	–	63.7	–	–	68.8	–	–	47.5	–	–
VideoGLaMM [59]	45.2	50.6	–	–	66.8	–	–	69.5	–	–	33.9	–	–
Sa2VA-8B [92]	46.9	57.0	–	–	**70.7**	–	–	75.2	–	–	55.5	–	–
Sa2VA-Qwen3-VL-4B [92]	36.7	57.1	–	–	68.1	–	–	**76.0**	–	–	50.0	–	–
Fully open													
Molmo [22] + SAM 2 [60]	46.9	51.5	53.8	–	64.6	71.1	–	65.2	74.5	–	45.7	50.3	–
VideoMolmo-7B [2]	53.9	57.0	59.4	–	67.3	73.7	–	72.5	75.4	–	51.1	50.3	–
Molmo2-4B	63.3	70.0	75.5	72.4	70.2	80.4	78.8	73.5	83.1	81.1	61.9	66.5	64.0
Molmo2-8B	62.3	70.8	75.9	72.6	70.2	78.7	77.3	72.7	81.3	78.7	**65.8**	**70.8**	**68.6**
Molmo2-O-7B	58.4	69.7	76.1	72.3	67.9	77.7	76.1	70.4	79.2	76.0	62.6	67.5	65.1
MolmoPoint													
MolmoPoint-8B	**63.5**	**72.2**	**77.0**	**73.8**	70.5	**81.9**	**80.5**	73.6	**84.1**	**82.2**	64.7	68.8	67.0

Table 6 Results on Molmo2Track benchmark by video domain. Overall reports the accuracy across all samples.

	Animals			Person			Sports			Dancers			Misc			Overall		
Model	$\mathcal{J}\&\mathcal{F}$	F1	HOTA	$\mathcal{J}\&\mathcal{F}$	F1	HOTA	$\mathcal{J}\&\mathcal{F}$	F1	HOTA	$\mathcal{J}\&\mathcal{F}$	F1	HOTA	$\mathcal{J}\&\mathcal{F}$	F1	HOTA	$\mathcal{J}\&\mathcal{F}$	F1	HOTA
API call only																		
GPT-5 [56]	41.4	20.6	20.3	16.5	4.5	4.2	14.4	2.0	2.5	33.8	11.7	11.5	14.6	2.2	1.6	23.5	7.5	7.5
GPT-5 mini [56]	21.7	7.8	8.0	8.6	1.6	1.5	10.7	0.6	0.8	15.6	2.1	2.0	13.5	0.6	0.4	12.7	2.1	2.1
Gemini 3 Pro [20]	70.4	62.3	60.0	44.5	30.7	29.2	23.4	10.3	8.8	55.6	44.3	37.8	35.3	18.3	14.4	44.6	32.2	29.1
Gemini 2.5 Pro [20]	69.3	56.8	53.2	50.0	33.6	31.9	29.7	10.8	8.9	55.9	39.4	32.2	34.7	17.6	18.3	47.9	31.2	27.8
Gemini 2.5 Flash [20]	58.0	46.6	44.4	38.9	21.4	20.1	13.2	6.2	5.5	48.0	29.0	25.1	21.9	5.7	4.6	36.2	21.8	19.8
Open models																		
Qwen3-VL-4B [84]	57.2	11.5	12.3	35.1	12.0	11.2	3.8	0.4	0.4	34.6	6.9	5.7	17.5	6.2	4.2	28.5	7.2	6.7
Qwen3-VL-8B [84]	63.8	52.3	50.2	35.4	20.3	18.9	5.2	1.7	1.4	31.3	19.0	16.7	16.3	6.2	4.2	28.7	18.0	16.5
Specialized open models																		
VideoLISA [7]	67.8	–	–	35.8	–	–	32.9	–	–	53.6	–	–	25.8	–	–	43.3	–	–
VideoGLaMM [59]	63.9	–	–	26.2	–	–	34.3	–	–	46.0	–	–	22.3	–	–	37.9	–	–
Sa2VA-8B [92]	74.3	–	–	45.5	–	–	30.7	–	–	53.3	–	–	**49.1**	–	–	46.9	–	–
Sa2VA-Qwen3-VL-4B [92]	73.3	–	–	48.6	–	–	31.6	–	–	50.1	–	–	31.4	–	–	46.7	–	–
SAM 3 [10]	41.1	–	–	35.2	–	–	43.3	–	–	29.2	–	–	36.8	–	–	36.3	–	–
Molmo [22] + SAM 2 [60]	71.8	76.0	–	52.7	7.0	–	52.8	2.6	–	51.7	7.55	–	40.9	37.5	–	54.2	14.0	–
VideoMolmo-7B [2]	68.4	69.5	–	51.1	6.3	–	43.2	2.1	–	53.8	7.2	–	39.9	30.8	–	51.3	12.7	–
Fully Open																		
Molmo2-4B	81.0	83.0	83.7	43.7	48.3	47.7	59.7	53.1	54.3	60.4	64.4	**64.4**	43.1	35.1	31.3	56.7	57.5	57.6
Molmo2-8B	80.1	82.0	83.0	43.1	47.9	48.0	59.8	53.3	54.8	59.9	63.9	63.5	41.6	31.5	29.7	56.2	57.1	57.5
Molmo2-O-7B	80.1	81.9	82.8	41.5	45.5	45.4	54.1	47.6	48.6	57.7	61.0	60.3	45.0	37.6	34.7	53.7	54.2	54.2
MolmoPoint																		
MolmoPoint-8B	**81.9**	**84.4**	**85.8**	**56.9**	**51.2**	**50.9**	**63.6**	**56.9**	**57.7**	**61.8**	**65.6**	64.2	**49.1**	**45.7**	**43.1**	**62.5**	**60.2**	**60.0**

Table 7 Tracking ablations on Molmo2Track. We consecutively remove grounding tokens and the newly introduced MolmoPoint-Track data.

Model	Animals		Person		Sports		Dancers		Misc		Overall	
	F1	HOTA	F1	HOTA	F1	HOTA	F1	HOTA	F1	HOTA	F1	HOTA
MolmoPoint-8B-Ablation	83.4	85.3	49.2	49.1	56.5	57.6	64.9	63.7	47.5	44.5	59.3	59.3
w/o grounding tokens	80.7	81.8	44.4	44.5	49.4	51.4	62.9	62.9	37.9	35.5	54.7	55.3
w/o MolmoPoint-Track	80.0	81.0	43.5	43.8	49.4	50.5	61.8	61.6	30.3	28.1	53.8	54.2

Removing grounding tokens leads to a **drop of 4.6 F1** and **4.0 HOTA** overall, suggesting grounding tokens are particularly beneficial for tracking a diverse set of object types. Further removing MolmoPoint-Track yields additional losses, most notably on Misc (**-7.6 F1**), confirming that the expanded data coverage addresses gaps where the original training data had limited representation.

5.5 MultiTask Results

To evaluate how well MolmoPoint-8B works as a general-purpose VLM, Table 8 reports average performance on 11 image benchmarks [33, 51, 52, 53, 63, 30, 78, 95, 47, 8, 22], 3 multi-image benchmarks [70, 54, 26], 6 short-video benchmarks [79, 57, 38, 62, 31, 43], and 7 long video benchmarks [25, 75, 101, 73, 49, 50] following the evaluation protocol Molmo2. Full details are in the appendix.

Overall, we see only small changes compared to Molmo2-8B: a small gain on images, a small loss on multi-images and short videos, and no difference on long videos. Since the data and training protocol are mostly unchanged between the two models, better transfer from image pointing data to image tasks might explain the gain in images. Although some interference between grounding and QA might likewise explain the drop in short videos. We do observe another sign of positive transfer during pre-training, where captioning performance is slightly better for MolmoPoint-8B after pre-training (55.6 vs 55.9 F1 on the Molmo captioning metric [22]). However, in either case, the effect is small.

5.6 Modeling Ablations

Modeling ablations are shown in Table 9. Due to limited compute resources, we do ablations with a lighter-weight training pipeline that starts from a pre-trained captioning model (without any pointing capabilities) and then fine-tunes on the image and video pointing data from the Molmo2 data mixture. We train for 6000 steps with a batch size of 64. For video input, we observe that models can produce degenerate outputs with an excessive number of points. To capture this, we add an *overcount* metric, defined as how often predictions are > 10 points and >= twice the number of ground truth points.

Removing rotary embeddings decreases performance a bit on images and has a more significant effect on video. Removing the no-more-points token hurts performance and more than doubles the amount of overcounting. Randomizing the order of the points shows a significant drop for video, but a surprising gain on PointBench. We hypothesize that this might improve performance by allowing models to generate points in an "easiest point first" order [12], but more work will be needed to take advantage of this without degrading video.

5.7 Sample Efficiency

Figure 5 (left) shows performance when fine-tuning a base captioning model on a small number of pointing examples. Models were tuned for 6 epochs to ensure they were fully saturated. MolmoPoint-8B initially performs worse, likely due to needing to learn new parameters from scratch, but quickly improves to a 20 point gain when using 8192 examples (the full pointing dataset has close to half a million examples). Figure 5 (right) shows MolmoPoint reaches peak performance faster during pre-training. Both of these results show that grounding tokens make pointing easier and more efficient to learn than text coordinates.

Table 8 Image and video QA results. Averaged results on the QA benchmarks in [19], see the appendix for details. Some results were computed by [19].

Model	Image Avg.	Multi-Image Avg.	Short Video Avg.	Long Video Avg.
API call only				
GPT-5[56]	83.7	72.1	73.1	76.3
GPT-5 mini[56]	81.9	68.2	66.8	69.8
Gemini 3 Pro[28]	86.2	81.9	71.0	78.8
Gemini 2.5 Pro[20]	81.3	72.4	71.1	80.4
Gemini 2.5 Flash[20]	79.3	68.3	67.0	74.5
Claude Sonnet 4.5[4]	76.3	59.5	62.8	66.4
Open weights				
InternVL3.5-4B[72]	77.2	53.5	62.0	56.5
InternVL3.5-8B[72]	78.2	54.9	63.0	57.1
Qwen3-VL-4B[6]	78.4	57.6	63.7	62.7
Qwen3-VL-8B[6]	81.2	56.3	65.3	63.5
Keye-VL-1.5-8B[85]	79.8	52.1	60.1	60.4
GLM-4.1V-9B[68]	77.0	**67.4**	64.2	60.5
MiniCPM-V-4.5-8B[91]	77.7	47.3	62.1	60.1
Eagle2.5-8B[13]	81.2	52.0	67.0	**65.2**
Fully open				
PLM-3B[18]	75.9	40.6	66.3	53.5
PLM-8B[18]	78.7	35.7	68.5	56.2
LLaVA-Video-7B[97]	-	-	59.4	56.2
VideoChat-Flash-7B[40]	-	-	66.4	58.1
Molmo2-4B[19]	80.4	57.8	69.3	64.5
Molmo2-8B[19]	81.7	56.4	**69.9**	64.1
Molmo2-O-7B[19]	79.7	53.5	68.1	59.2
MolmoPoint				
MolmoPoint-8B	**82.2**	56.0	69.5	64.2

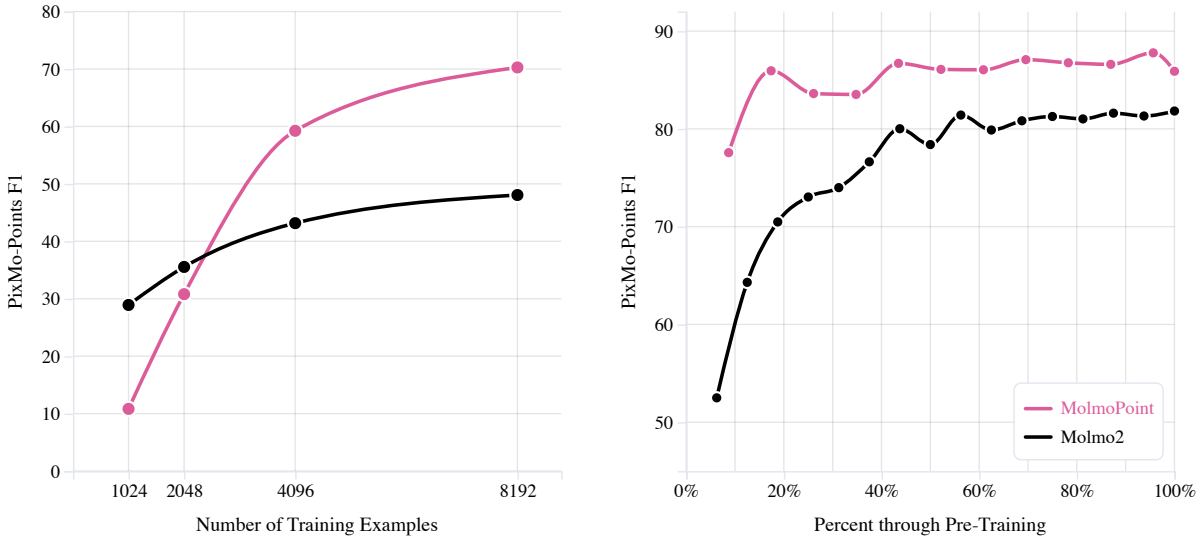
Table 9 Ablations. Results when removing rotary embeddings, the no-more-points class, or the constraint that the points must be generated in sorted order.

Model	PixMoPoint	PointBench	Molmo2-VC		
	F1	Avg.	Correct	Close	Overcount ↓
Molmo2-P-Ablation	85.2	67.8	58.0	36.6	3.6
w/o rotary	84.5	67.6	56.8	33.0	4.5
w/o no-more-points	84.7	66.6	52.3	32.8	10.3
w/o point sorting	83.6	71.2	40.0	24.4	3.2

5.8 Qualitative results

Despite being trained on the same data, we observe significant qualitative differences between MolmoPoint-8B and Molmo2-8B. MolmoPoint-8B is less likely to produce degenerate output on videos, is better at finding small objects, and can be more precise when pointing. However, we observe that it occasionally produces off-by-one errors when counting high-frequency objects. Figure 6 demonstrates qualitative examples to compare Molmo2 and Molmo2-P in video pointing, high-resolution GUI grounding, and multi-object pointing.

Figure 5 Sample efficiency. Left: Performance when using a very limited number of pointing training examples. Right: Pointing performance during full-scale pre-training.



6 Conclusion

We have shown that using grounding tokens significantly improves pointing across multiple domains. The improvements in sample efficiency and training speed show this method would be especially helpful in low-resource settings. Future work could extend this approach to include other modalities, such as pointing to text tokens to highlight important parts of the text or pointing to audio tokens to reference a sound.

Acknowledgements

This work would not be possible without the support of our colleagues at Ai2.

- We thank David Albright, Cailin Brashear, Crystal Nam, Kyle Wiggers, and Will Smith for their important work for the MolmoPoint-8B public release.
- We thank other members of the PRIOR team for providing advice and feedback on various aspects of MolmoPoint-8B.
- We thank the Prolific team for their support and our annotators on Prolific for providing us with high-quality data that is crucial to MolmoPoint-8B.

This material is based upon work supported by the National Science Foundation under Award No. 2413244.

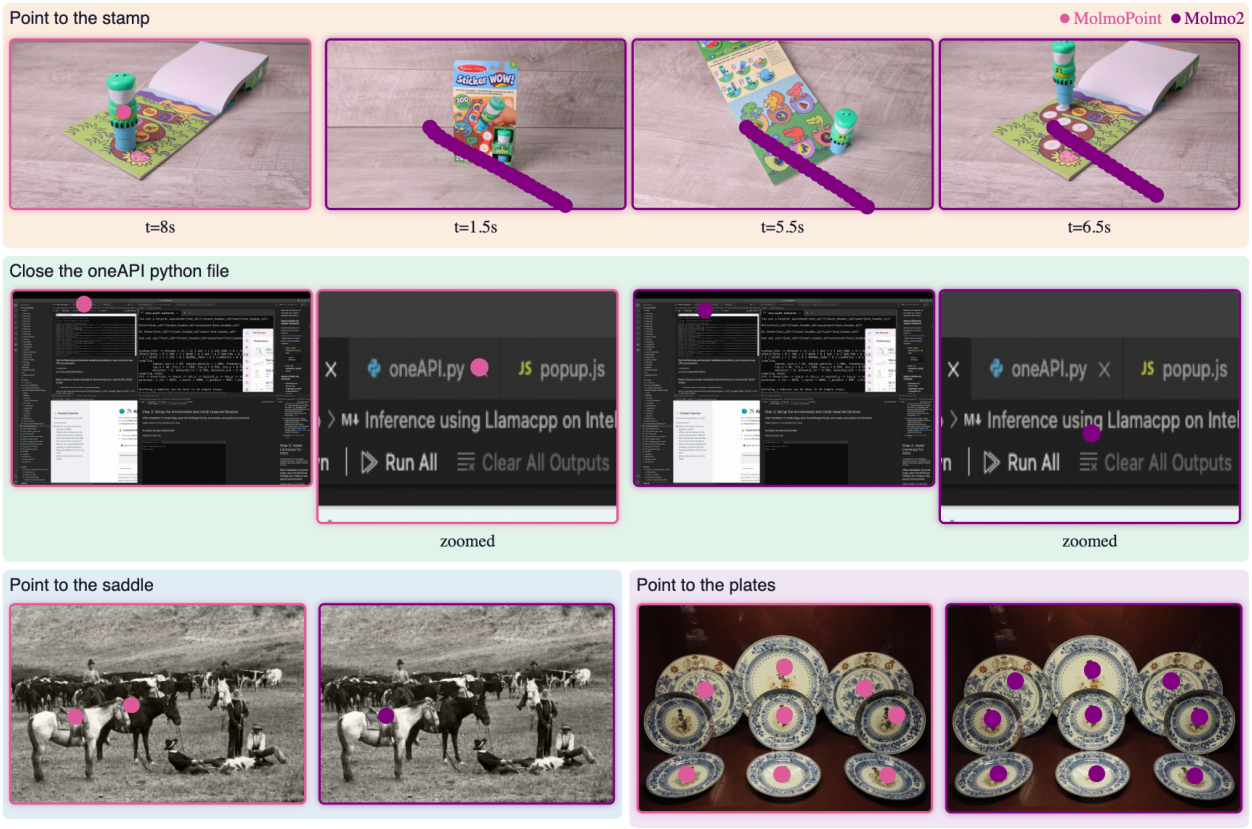


Figure 6 Qualitative examples. Top, Molmo2 generates lines of incorrect points in multiple video frames. Middle, Molmo2 is unable to localize the "x" exactly (zoomed images are close-ups of the same point). Bottom left, MolmoPoint-8B finds the second, partly occluded saddle. Bottom right, MolmoPoint-8B misses one of the plates. The middle row shows results from MolmoPoint-GUI-8B and Molmo2-GUI-8B, and the others show MolmoPoint-8B and Molmo2-8B.

References

[1] A. Abdolmaleki, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, A. Balakrishna, N. Batchelor, A. Bewley, J. Bingham, M. Bloesch, et al. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer. *arXiv preprint arXiv:2510.03342*, 2025.

[2] G. S. Ahmad, A. Heakl, H. Gani, A. Shaker, Z. Shen, F. S. Khan, and S. Khan. Videomolmo: Spatio-temporal grounding meets pointing. *arXiv preprint arXiv:2506.05336*, 2025.

[3] Anthropic. The claude 3 model family: Opus, sonnet, haiku, 2024. URL https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf.

[4] Anthropic. Claude sonnet 4.5 system card, 2025. URL https://assets.anthropic.com/m/12f214efcc2f457a/original/Claude-Sonnet-4-5-System-Card.pdf.

[5] A. Athar, J. Luiten, P. Voigtlaender, T. Khurana, A. Dave, B. Leibe, and D. Ramanan. Burst: A benchmark for unifying object recognition, segmentation and tracking in video. In *WACV*, 2023.

[6] S. Bai, Y. Cai, R. Chen, K. Chen, X. Chen, Z. Cheng, L. Deng, W. Ding, C. Gao, C. Ge, W. Ge, Z. Guo, Q. Huang, J. Huang, F. Huang, B. Hui, S. Jiang, Z. Li, M. Li, M. Li, K. Li, Z. Lin, J. Lin, X. Liu, J. Liu, C. Liu, Y. Liu, D. Liu, S. Liu, D. Lu, R. Luo, C. Lv, R. Men, L. Meng, X. Ren, X. Ren, S. Song, Y. Sun, J. Tang, J. Tu, J. Wan, P. Wang, P. Wang, Q. Wang, Y. Wang, T. Xie, Y. Xu, H. Xu, J. Xu, Z. Yang, M. Yang, J. Yang, A. Yang, B. Yu, F. Zhang, H. Zhang, X. Zhang, B. Zheng, H. Zhong, J. Zhou, F. Zhou, J. Zhou, Y. Zhu, and K. Zhu. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[7] Z. Bai, T. He, H. Mei, P. Wang, Z. Gao, J. Chen, L. Liu, Z. Zhang, and M. Z. Shou. One token to seg them all: Language instructed reasoning segmentation in videos. In *NeurIPS*, 2024.

[8] L. Beyer, A. Steiner, A. S. Pinto, A. Kolesnikov, X. Wang, D. Salz, M. Neumann, I. Alabdulmohsin, M. Tschannen, E. Bugliarello, T. Unterthiner, D. Keysers, S. Koppula, F. Liu, A. Grycner, A. Gritsenko, N. Houlsby, M. Kumar, K. Rong, J. Eisenschlos, R. Kabra, M. Bauer, M. Bošnjak, X. Chen, M. Minderer, P. Voigtlaender, I. Bica, I. Balazevic, J. Puigcerver, P. Papalampidi, O. Henaff, X. Xiong, R. Soricut, J. Harmsen, and X. Zhai. PaliGemma: A versatile 3B VLM for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

[9] M. Bigverdi, Z. Luo, C.-Y. Hsieh, E. Shen, D. Chen, L. G. Shapiro, and R. Krishna. Perception tokens enhance visual reasoning in multimodal language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 3836–3845, 2025.

[10] N. Carion, L. Gustafson, Y.-T. Hu, S. Debnath, R. Hu, D. Suris, C. Ryali, K. V. Alwala, H. Khedr, A. Huang, et al. Sam 3: Segment anything with concepts. *arXiv preprint arXiv:2511.16719*, 2025.

[11] Y. Chai, S. Huang, Y. Niu, H. Xiao, L. Liu, D. Zhang, P. Gao, S. Ren, and H. Li. Amex: Android multi-annotation expo dataset for mobile gui agents, 2024. URL https://arxiv.org/abs/2407.17490.

[12] H. Chang, H. Zhang, L. Jiang, C. Liu, and W. T. Freeman. Maskgit: Masked generative image transformer. In *CVPR*, 2022.

[13] G. Chen, Z. Li, S. Wang, J. Jiang, Y. Liu, L. Lu, D.-A. Huang, W. Byeon, M. Le, M. Ehrlich, T. Lu, L. Wang, B. Catanzaro, J. Kautz, A. Tao, Z. Yu, and G. Liu. Eagle 2.5: Boosting long-context post-training for frontier vision-language models. In *NeurIPS*, 2025.

[14] T. Chen, S. Saxena, L. Li, D. J. Fleet, and G. Hinton. Pix2seq: A language modeling framework for object detection. *arXiv preprint arXiv:2109.10852*, 2021.

[15] K. Cheng, Q. Sun, Y. Chu, F. Xu, L. YanTao, J. Zhang, and Z. Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9313–9332, 2024.

[16] L. Cheng, J. Duan, Y. R. Wang, H. Fang, B. Li, Y. Huang, E. Wang, A. Eftekhar, J. Lee, W. Yuan, et al. Pointarena: Probing multimodal grounding through language-guided pointing. *arXiv preprint arXiv:2505.09990*, 2025.

[17] W.-L. Chiang, L. Zheng, Y. Sheng, A. N. Angelopoulos, T. Li, D. Li, H. Zhang, B. Zhu, M. Jordan, J. E. Gonzalez, and I. Stoica. Chatbot arena: An open platform for evaluating LLMs by human preference. In *ICML*, 2024.

[18] J. H. Cho, A. Madotto, E. Mavroudi, T. Afouras, T. Nagarajan, M. Maaz, Y. Song, T. Ma, S. Hu, H. Rasheed, P. Sun, P.-Y. Huang, D. Bolya, S. Jain, M. Martin, H. Wang, N. Ravi, S. Jain, T. Stark, S. Moon, B. Damavandi, V. Lee, A. Westbury, S. Khan, P. Krähenbühl, P. Dollár, L. Torresani, K. Grauman, and C. Feichtenhofer. Perceptionlm: Open-access data and models for detailed visual understanding. *arXiv preprint arXiv:2504.13180*, 2025.

[19] C. Clark, J. Zhang, Z. Ma, J. S. Park, M. Salehi, R. Tripathi, S. Lee, Z. Ren, C. D. Kim, Y. Yang, et al. Molmo2: Open weights and data for vision-language models with video understanding and grounding. *arXiv preprint arXiv:2601.10611*, 2026.

[20] G. Comanici, E. Bieber, M. Schaekermann, I. Pasupat, N. Sachdeva, I. Dhillon, M. Blistein, O. Ram, D. Zhang, E. Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

[21] H. Company. Holo2 - open foundation models for navigation and computer use agents, 2025. URL https://huggingface.co/collections/Hcompany/holo2.

[22] M. Deitke, C. Clark, S. Lee, R. Tripathi, Y. Yang, J. S. Park, M. Salehi, N. Muennighoff, K. Lo, L. Soldaini, J. Lu, T. Anderson, E. Bransom, K. Ehsani, H. Ngo, Y. Chen, A. Patel, M. Yatskar, C. Callison-Burch, A. Head, R. Hendrix, F. Bastani, E. VanderBilt, N. Lambert, Y. Chou, A. Chheda, J. Sparks, S. Skjonsberg, M. Schmitz, A. Sarnat, B. Bischoff, P. Walsh, C. Newell, P. Wolters, T. Gupta, K.-H. Zeng, J. Borchardt, D. Groeneveld, C. Nam, S. Lebrecht, C. Wittlif, C. Schoenick, O. Michel, R. Krishna, L. Weihs, N. A. Smith, H. Hajishirzi, R. Girshick, A. Farhadi, and A. Kembhavi. Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models. In *CVPR*, 2025.

[23] H. Ding, C. Liu, S. He, X. Jiang, and C. C. Loy. Mevis: A large-scale benchmark for video segmentation with motion expressions. In *ICCV*, 2023.

[24] A. Feizi, S. Nayak, X. Jian, K. Q. Lin, K. Li, R. Awal, X. H. Lù, J. Obando-Ceron, J. A. Rodriguez, N. Chapados, D. Vazquez, A. Romero-Soriano, R. Rabbany, P. Taslakian, C. Pal, S. Gella, and S. Rajeswar. Grounding computer use agents on human demonstrations. *arXiv preprint arXiv:2511.07332*, 2025.

[25] C. Fu, Y. Dai, Y. Luo, L. Li, S. Ren, R. Zhang, Z. Wang, C. Zhou, Y. Shen, M. Zhang, et al. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In *CVPR*, 2025.

[26] X. Fu, Y. Hu, B. Li, Y. Feng, H. Wang, X. Lin, D. Roth, N. A. Smith, W.-C. Ma, and R. Krishna. Blink: Multimodal large language models can see but not perceive. In *ECCV*, 2024.

[27] T. Ge, X. Chan, X. Wang, D. Yu, H. Mi, and D. Yu. Scaling synthetic data creation with 1,000,000,000 personas. *arXiv preprint arXiv:2406.20094*, 2024.

[28] Google. Gemini 3 Pro model card, 2025. URL https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Pro-Model-Card.pdf.

[29] B. Gou, R. Wang, B. Zheng, Y. Xie, C. Chang, Y. Shu, H. Sun, and Y. Su. Navigating the digital world as humans do: Universal visual grounding for GUI agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=kxnoqaisCT.

[30] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *CVPR*, 2017.

[31] W. Hong, Y. Cheng, Z. Yang, W. Wang, L. Wang, X. Gu, S. Huang, Y. Dong, and J. Tang. Motionbench: Benchmarking and improving fine-grained video motion understanding for vision language models. In *CVPR*, 2025.

[32] R. Kapoor, Y. P. Butala, M. Russak, J. Y. Koh, K. Kamble, W. AlShikh, and R. Salakhutdinov. Omniact: A dataset and benchmark for enabling multimodal generalist autonomous agents for desktop and web. In *European Conference on Computer Vision*, pages 161–178. Springer, 2024.

[33] A. Kembhavi, M. Salvato, E. Kolve, M. Seo, H. Hajishirzi, and A. Farhadi. A diagram is worth a dozen images. In *ECCV*, 2016.

[34] A. Khoreva, A. Rohrbach, and B. Schiele. Video object segmentation with language referring expressions. In *ACCV*, 2018.

[35] X. Lai, Z. Tian, Y. Chen, Y. Li, Y. Yuan, S. Liu, and J. Jia. Lisa: Reasoning segmentation via large language model. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9579–9589, 2024.

[36] J. Lee, J. Duan, H. Fang, Y. Deng, S. Liu, B. Li, B. Fang, J. Zhang, Y. R. Wang, S. Lee, W. Han, W. Pumacay, A. Wu, R. Hendrix, K. Farley, E. VanderBilt, A. Farhadi, D. Fox, and R. Krishna. Molmoact: Action reasoning models that can reason in space. *arXiv preprint arXiv:2508.07917*, 2025.

[37] J. Lee, J. Duan, H. Fang, Y. Deng, S. Liu, B. Li, B. Fang, J. Zhang, Y. R. Wang, S. Lee, et al. Molmoact: Action reasoning models that can reason in space. *arXiv preprint arXiv:2508.07917*, 2025.

[38] K. Li, Y. Wang, Y. He, Y. Li, Y. Wang, Y. Liu, Z. Wang, J. Xu, G. Chen, P. Luo, et al. Mvbench: A comprehensive multi-modal video understanding benchmark. In *CVPR*, 2024.

[39] K. Li, Z. Meng, H. Lin, Z. Luo, Y. Tian, J. Ma, Z. Huang, and T.-S. Chua. Screenspot-pro: Gui grounding for professional high-resolution computer use. In *MM*, 2025.

[40] X. Li, Y. Wang, J. Yu, X. Zeng, Y. Zhu, H. Huang, J. Gao, K. Li, Y. He, C. Wang, Y. Qiao, Y. Wang, and L. Wang. Videochat-flash: Hierarchical compression for long-context video modeling. *arXiv preprint arXiv:2501.00574*, 2024.

[41] Y. Li, J. Zhang, X. Teng, H. Zhang, X. Liu, and L. Lan. Refsam: Efficiently adapting segmenting anything model for referring video object segmentation. *Neural Networks*, 2025.

[42] K. Q. Lin, L. Li, D. Gao, Z. Yang, S. Wu, Z. Bai, W. Lei, L. Wang, and M. Z. Shou. Showui: One vision-language-action model for gui visual agent, 2024. URL https://arxiv.org/abs/2411.17465.

[43] Y. Liu, S. Li, Y. Liu, Y. Wang, S. Ren, L. Li, S. Chen, X. Sun, and L. Hou. Tempcompass: Do video llms really understand videos? In *ACL*, 2024.

[44] Y. Liu, T. Qu, Z. Zhong, B. Peng, S. Liu, B. Yu, and J. Jia. Visionreasoner: Unified visual perception and reasoning via reinforcement learning. *arXiv preprint arXiv:2505.12081*, 2025.

[45] J. Lu, C. Clark, R. Zellers, R. Mottaghi, and A. Kembhavi. Unified-io: A unified model for vision, language, and multi-modal tasks. *arXiv preprint arXiv:2206.08916*, 2022.

[46] J. Lu, C. Clark, S. Lee, Z. Zhang, S. Khosla, R. Marten, D. Hoiem, and A. Kembhavi. Unified-io 2: Scaling autoregressive multimodal models with vision language audio and action. In *CVPR*, 2024.

[47] P. Lu, H. Bansal, T. Xia, J. Liu, C. Li, H. Hajishirzi, H. Cheng, K.-W. Chang, M. Galley, and J. Gao. MathVista: Evaluating mathematical reasoning of foundation models in visual contexts. In *ICLR*, 2024.

[48] J. Luiten, A. Osep, P. Dendorfer, P. Torr, A. Geiger, L. Leal-Taixé, and B. Leibe. Hota: A higher order metric for evaluating multi-object tracking. *IJCV*, 2021.

[49] W. Ma, W. Ren, Y. Jia, Z. Li, P. Nie, G. Zhang, and W. Chen. Videoeval-pro: Robust and realistic long video understanding evaluation. *arXiv preprint arXiv:2505.14640*, 2025.

[50] K. Mangalam, R. Akshulakov, and J. Malik. Egoschema: A diagnostic benchmark for very long-form video language understanding. In *NeurIPS Track on Datasets and Benchmarks*, 2023.

[51] A. Masry, D. Long, J. Q. Tan, S. Joty, and E. Hoque. ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In *ACL*, 2022.

[52] M. Mathew, D. Karatzas, and C. Jawahar. DocVQA: A dataset for VQA on document images. In *WACV*, 2021.

[53] M. Mathew, V. Bagal, R. Tito, D. Karatzas, E. Valveny, and C. Jawahar. InfographicVQA. In *WACV*, 2022.

[54] F. Meng, J. Wang, C. Li, Q. Lu, H. Tian, J. Liao, X. Zhu, J. Dai, Y. Qiao, P. Luo, K. Zhang, and W. Shao. Mmiu: Multimodal multi-image understanding for evaluating large vision-language models. In *ICLR*, 2025.

[55] OpenAI. Collaborative user agreement, 2024. URL https://openai.com/policies/service-terms/. Accessed: 2025-03-01.

[56] OpenAI. GPT-5 system card, 2025. URL https://openai.com/index/gpt-5-system-card/.

[57] V. Patraucean, L. Smaira, A. Gupta, A. Recasens, L. Markeeva, D. Banarse, S. Koppula, M. Malinowski, Y. Yang, C. Doersch, et al. Perception test: A diagnostic benchmark for multimodal video models. *NeurIPS*, 2023.

[58] Y. Qin, Y. Ye, J. Fang, H. Wang, S. Liang, S. Tian, J. Zhang, J. Li, Y. Li, S. Huang, et al. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025.

[59] H. Rasheed, M. Maaz, S. Shaji, A. Shaker, S. Khan, H. Cholakkal, R. M. Anwer, E. Xing, M.-H. Yang, and F. S. Khan. Glamm: Pixel grounding large multimodal model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13009–13018, 2024.

[60] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer. Sam 2: Segment anything in images and videos. In *ICLR*, 2025.

[61] S. Seo, J.-Y. Lee, and B. Han. Urvos: Unified referring video object segmentation network with a large-scale benchmark. In *ECCV*, 2020.

[62] Z. Shangguan, C. Li, Y. Ding, Y. Zheng, Y. Zhao, T. Fitzgerald, and A. Cohan. Tomato: Assessing visual temporal reasoning capabilities in multimodal foundation models. In *ICLR*, 2025.

[63] A. Singh, V. Natarjan, M. Shah, Y. Jiang, X. Chen, D. Parikh, and M. Rohrbach. Towards VQA models that can read. In *CVPR*, 2019.

[64] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 2024.

[65] Y. Su, H. Zhang, S. Li, N. Liu, J. Liao, J. Pan, Y. Liu, X. Xing, C. Sun, C. Li, et al. Patch-as-decodable-token: Towards unified multi-modal vision tasks in mllms. *arXiv preprint arXiv:2510.01954*, 2025.

[66] Q. Sun, P. Hong, T. D. Pala, V. Toh, U.-X. Tan, D. Ghosal, and S. Poria. Emma-x: An embodied multimodal action model with grounded chain of thought and look-ahead spatial reasoning. In *ACL*, 2025.

[67] J. Tang, Y. Xia, Y.-F. Wu, Y. Hu, Y. Chen, Q. Chen, X. Xu, X. Wu, H. Lu, Y. Ma, S. Lu, and Q. Chen. Lpo: Towards accurate gui agent interaction via location preference optimization. *ArXiv*, abs/2506.09373, 2025. URL https://api.semanticscholar.org/CorpusID:279306118.

[68] V. Team, W. Hong, W. Yu, X. Gu, G. Wang, G. Gan, H. Tang, J. Cheng, J. Qi, J. Ji, L. Pan, S. Duan, W. Wang, Y. Wang, Y. Cheng, Z. He, Z. Su, Z. Yang, Z. Pan, A. Zeng, B. Wang, B. Chen, B. Shi, C. Pang, C. Zhang, D. Yin, F. Yang, G. Chen, J. Xu, J. Zhu, J. Chen, J. Chen, J. Chen, J. Lin, J. Wang, J. Chen, L. Lei, L. Gong, L. Pan, M. Liu, M. Xu, M. Zhang, Q. Zheng, S. Yang, S. Zhong, S. Huang, S. Zhao, S. Xue, S. Tu, S. Meng, T. Zhang, T. Luo, T. Hao, T. Tong, W. Li, W. Jia, X. Liu, X. Zhang, X. Lyu, X. Fan, X. Huang, Y. Wang, Y. Xue, Y. Wang, Y. Wang, Y. An, Y. Du, Y. Shi, Y. Huang, Y. Niu, Y. Wang, Y. Yue, Y. Li, Y. Zhang, Y. Wang, Y. Wang, Y. Zhang, Z. Xue, Z. Hou, Z. Du, Z. Wang, P. Zhang, D. Liu, B. Xu, J. Li, M. Huang, Y. Dong, and J. Tang. Glm-4.5v and glm-4.1v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning. *arXiv preprint arXiv:2507.01006*, 2025.

[69] V. Team, C. Gao, Z. Gu, Y. Liu, X. Qiu, S. Shen, Y. Wen, T. Xia, Z. Xu, Z. Zeng, B. Zhou, X. Zhou, W. Chen, S. Dai, J. Dou, Y. Gong, Y. Guo, Z. Guo, F. Li, Q. Li, J. Lin, Y. Zhou, L. Zhu, L. Chen, Z. Guo, C. Meng, and W. Wang. Ui-venus-1.5 technical report. *arXiv preprint arXiv:2602.09082*, 2026.

[70] F. Wang, X. Fu, J. Y. Huang, Z. Li, Q. Liu, X. Liu, M. D. Ma, N. Xu, W. Zhou, K. Zhang, et al. Muirbench: A comprehensive benchmark for robust multi-image understanding. In *ICLR*, 2025.

[71] P. Wang, A. Yang, R. Men, J. Lin, S. Bai, Z. Li, J. Ma, C. Zhou, J. Zhou, and H. Yang. Ofa: Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. In *International conference on machine learning*, pages 23318–23340. PMLR, 2022.

[72] W. Wang, Z. Gao, L. Gu, H. Pu, L. Cui, X. Wei, Z. Liu, L. Jing, S. Ye, J. Shao, et al. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025.

[73] W. Wang, Z. He, W. Hong, Y. Cheng, X. Zhang, J. Qi, M. Ding, X. Gu, S. Huang, B. Xu, et al. Lvbench: An extreme long video understanding benchmark. In *ICCV*, 2025.

[74] X. Wang, B. Wang, D. Lu, J. Yang, T. Xie, J. Wang, J. Deng, X. Guo, Y. Xu, C. H. Wu, Z. Shen, Z. Li, R. Li, X. Li, J. Chen, B. Zheng, P. Li, F. Lei, R. Cao, Y. Fu, D. Shin, M. Shin, J. Hu, Y. Wang, J. Chen, Y. Ye, D. Zhang, D. Du, H. Hu, H. Chen, Z. Zhou, H. Yao, Z. Chen, Q. Gu, Y. Wang, H. Wang, D. Yang, V. Zhong, F. Sung, Y. Charles, Z. Yang, and T. Yu. Opencua: Open foundations for computer-use agents. *arXiv preprint arXiv:2508.09123*, 2025.

[75] H. Wu, D. Li, B. Chen, and J. Li. Longvideobench: A benchmark for long-context interleaved video-language understanding. In *NeurIPS*, 2024.

[76] Q. Wu, K. Cheng, R. Yang, C. Zhang, J. Yang, H. Jiang, J. Mu, B. Peng, B. Qiao, R. Tan, et al. Gui-actor: Coordinate-free visual grounding for gui agents. *arXiv preprint arXiv:2506.03143*, 2025.

[77] Z. Wu, Z. Wu, F. Xu, Y. Wang, Q. Sun, C. Jia, K. Cheng, Z. Ding, L. Chen, P. P. Liang, et al. Os-atlas: A foundation action model for generalist gui agents. *arXiv preprint arXiv:2410.23218*, 2024.

[78] xAI. RealWorldQA. https://huggingface.co/datasets/xai-org/RealworldQA, 2024. Accessed: 2024-09-24.

[79] J. Xiao, X. Shang, A. Yao, and T.-S. Chua. Next-qa: Next phase of question-answering to explaining temporal actions. In *CVPR*, 2021.

[80] T. Xie, J. Deng, X. Li, J. Yang, H. Wu, J. Chen, W. Hu, X. Wang, Y. Xu, Z. Wang, Y. Xu, J. Wang, D. Sahoo, T. Yu, and C. Xiong. Scaling computer-use grounding via user interface decomposition and synthesis. *arXiv preprint arXiv:2505.13227*, 2025.

[81] T. Xie, J. Deng, X. Li, J. Yang, H. Wu, J. Chen, W. Hu, X. Wang, Y. Xu, Z. Wang, Y. Xu, J. Wang, D. Sahoo, T. Yu, and C. Xiong. Scaling computer-use grounding via user interface decomposition and synthesis. *arXiv preprint arXiv:2505.13227*, 2025.

[82] C. Yan, H. Wang, S. Yan, X. Jiang, Y. Hu, G. Kang, W. Xie, and E. Gavves. Visa: Reasoning video object segmentation via large language models. In *ECCV*, 2024.

[83] A. Yang, B. Yang, B. Hui, B. Zheng, B. Yu, C. Zhou, C. Li, C. Li, D. Liu, F. Huang, G. Dong, H. Wei, H. Lin, J. Tang, J. Wang, J. Yang, J. Tu, J. Zhang, J. Ma, J. Xu, J. Zhou, J. Bai, J. He, J. Lin, K. Dang, K. Lu, K. Chen, K. Yang, M. Li, M. Xue, N. Ni, P. Zhang, P. Wang, R. Peng, R. Men, R. Gao, R. Lin, S. Wang, S. Bai, S. Tan, T. Zhu, T. Li, T. Liu, W. Ge, X. Deng, X. Zhou, X. Ren, X. Zhang, X. Wei, X. Ren, Y. Fan, Y. Yao, Y. Zhang, Y. Wan, Y. Chu, Y. Liu, Z. Cui, Z. Zhang, and Z. Fan. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

[84] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, C. Zheng, D. Liu, F. Zhou, F. Huang, F. Hu, H. Ge, H. Wei, H. Lin, J. Tang, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Zhou, J. Lin, K. Dang, K. Bao, K. Yang, L. Yu, L. Deng, M. Li, M. Xue, M. Li, P. Zhang, P. Wang, Q. Zhu, R. Men, R. Gao, S. Liu, S. Luo, T. Li, T. Tang, W. Yin, X. Ren, X. Wang, X. Zhang, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Zhang, Y. Wan, Y. Liu, Z. Wang, Z. Cui, Z. Zhang, Z. Zhou, and Z. Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[85] B. Yang, B. Wen, B. Ding, C. Liu, C. Chu, C. Song, C. Rao, C. Yi, D. Li, D. Zang, et al. Kwai keye-vl 1.5 technical report. *arXiv preprint arXiv:2509.01563*, 2025.

[86] J. Yang, C. K. Fu, D. Shah, D. Sadigh, F. Xia, and T. Zhang. Bridging perception and action: Spatially-grounded mid-level representations for robot generalization. *arXiv preprint arXiv:2506.06196*, 2025.

[87] W. Yang and Z. Huang. Poivre: Self-refining visual pointing with reinforcement learning. *arXiv preprint arXiv:2509.23746*, 2025.

[88] Y. Yang, Y. Wang, D. Li, Z. Luo, B. Chen, C. Huang, and J. Li. Aria-ui: Visual grounding for gui instructions. *arXiv preprint arXiv:2412.16256*, 2024.

[89] Y. Yang, D. Li, Y. Yang, Z. Luo, Y. Dai, Z. Chen, R. Xu, L. Pan, C. Xiong, and J. Li. Grpo for gui grounding done right, 2025. URL https://huggingface.co/HelloKKMe/GTA1-7B.

[90] Y. Yang, A. Patel, M. Deitke, T. Gupta, L. Weihs, A. Head, M. Yatskar, C. Callison-Burch, R. Krishna, A. Kembhavi, et al. Scaling text-rich image understanding via code-guided synthetic multimodal data generation. In *ACL*, 2025.

[91] T. Yu, Z. Wang, C. Wang, F. Huang, W. Ma, Z. He, T. Cai, W. Chen, Y. Huang, Y. Zhao, B. Xu, J. Cui, Y. Xu, L. Ruan, L. Zhang, H. Liu, J. Tang, H. Liu, Q. Guo, W. Hu, B. He, J. Zhou, J. Cai, J. Qi, Z. Guo, C. Chen, G. Zeng, Y. Li, G. Cui, N. Ding, X. Han, Y. Yao, Z. Liu, and M. Sun. Minicpm-v 4.5: Cooking efficient mllms via architecture, data, and training recipe. *arXiv preprint arXiv:2509.18154*, 2025.

[92] H. Yuan, X. Li, T. Zhang, Z. Huang, S. Xu, S. Ji, Y. Tong, L. Qi, J. Feng, and M.-H. Yang. Sa2va: Marrying sam2 with llava for dense grounded understanding of images and videos. *arXiv preprint arXiv:2501.04001*, 2025.

[93] W. Yuan, J. Duan, V. Blukis, W. Pumacay, R. Krishna, A. Murali, A. Mousavian, and D. Fox. Robopoint: A vision-language model for spatial affordance prediction for robotics. In *CoRL*, 2024.

[94] X. Yuan, J. Zhang, K. Li, Z. Cai, L. Yao, J. Chen, E. Wang, Q. Hou, J. Chen, P.-T. Jiang, and B. Li. Enhancing visual grounding for gui agents via self-evolutionary reinforcement learning. *ArXiv*, abs/2505.12370, 2025. URL https://api.semanticscholar.org/CorpusID:278739769.

[95] X. Yue, Y. Ni, K. Zhang, T. Zheng, R. Liu, G. Zhang, S. Stevens, D. Jiang, W. Ren, Y. Sun, C. Wei, B. Yu, R. Yuan, R. Sun, M. Yin, B. Zheng, Z. Yang, Y. Liu, W. Huang, H. Sun, Y. Su, and W. Chen. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *CVPR*, 2024.

[96] B. Zhang, B. Zhang, B. Wang, W. Zheng, Y. Cheng, L. Tang, Y. Yan, J. Zhou, and J. Lu. Manicog: Training-free improvement for gui grounding via manipulation chains. 2026.

[97] Y. Zhang, J. Wu, W. Li, B. Li, Z. Ma, Z. Liu, and C. Li. Llava-video: Video instruction tuning with synthetic data. *TMLR*, 2025.

[98] Y. Zhang, L. Zhang, R. Ma, and N. Cao. Texverse: A universe of 3d objects with high-resolution textures. *arXiv preprint arXiv:2508.10868*, 2025.

[99] Y. Zhao, A. Gu, R. Varma, L. Luo, C.-C. Huang, M. Xu, L. Wright, H. Shojanazeri, M. Ott, S. Shleifer, et al. Pytorch fsdp: Experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023.

[100] H. Zhou, X. Zhang, P. Tong, J. Zhang, L. Chen, Q. Kong, C. Cai, C. Liu, Y. Wang, J. Zhou, and S. Hoi. Mai-ui technical report: Real-world centric foundation gui agents. *arXiv preprint arXiv:2512.22047*, 2025.

[101] J. Zhou, Y. Shu, B. Zhao, B. Wu, Z. Liang, S. Xiao, M. Qin, X. Yang, Y. Xiong, B. Zhang, et al. Mlvu: Benchmarking multi-task long video understanding. In *CVPR*, 2025.

Table 10 Updated top-level sampling rates. For MolmoPoint-8B we slightly reduce the sampling rates on pointing tasks since we observe faster convergence of these tasks. Each of these categories consists of its own sub-mixture of datasets. We leave the sampling ratios within those submixtures unchanged. See [19] for the full list.

Dataset Group	MolmoPoint-8B	Molmo2
Captioning/Long QA	15.0	13.6
Image QA	25.0	22.7
Video QA	20.0	18.2
Image Pointing	7.0	9.1
Video Pointing	11.0	13.6
Video Tracking	12.0	13.6
NLP	10.0	9.1

Appendix

This appendix includes the following sections:

7 Training Details

Our training pipeline largely follows Molmo2, with a few differences to account for the hardware we have available, improve efficiency, and take advantage of the increased learning speed when using grounding tokens. We also expand the tracking data from Molmo2 to improve tracking robustness. We discuss the training changes below, see Section 11 for details on the tracking data.

Pre-training. We improve packing efficiency by allowing up to 16 images per input sequence. We reduce the total number of training steps from 32k to 23k to keep the number of examples seen unchanged. We use a learning rate of 1e−4 with a warmup of 200 for the pointing parameters.

SFT. We slightly tweak the mixture rates of Molmo2 because preliminary experiments showed that grounding tasks converged significantly faster when using grounding tokens. We do this by adjusting the top-level sampling rates used in the Molmo2 training pipeline; see Table 10. We also train for 22k steps with a batch size of 160 instead of 30k steps with a batch size of 128, which reduces the number of examples seen by about 8.3%. We still use a learning rate of 1e−4 for the pointing parameters.

Long-context SFT. We train with a batch size of 160 instead of 128, and we train with a maximum of 384 frames and 16384 tokens per training example, following Molmo2.

Specialized models. Both MolmoPoint-Vid-8B and MolmoPoint-GUI-8B used the same optimizer settings as the SFT model. We train MolmoPoint-GUI-8B for 2000 steps with a batch size of 128 and a maximum of 48 crops per image on MolmoPoint-GUISyn. We train MolmoPoint-Vid-8B using the same pre-training stage, and then training on just the Video Point dataset group from Molmo2 for 6000 steps and a batch size of 64, and then for another 800 steps with a maximum of 384 frames.

Training times . Full training times are shown in Table 11. All training was done on B200 GPUs. Following Molmo2, we use PyTorch with Torch's Fully Sharded Data Parallel (FSDP) 2 [99] for distributed training

Table 11 Training times. Columns show the model, the training phase, gpu counts, training hours, total GPU hours, batch size, the estimated number of multi-modal inputs that were seen (separately counting inputs that were packed together), and the estimated number of training annotations that were seen (separately counting annotations that were merged into message trees).

Model	phase	Hardware	GPUs	hours	GPU hr.	batch	steps	mm.	annotations
MolmoPoint-8B	Pretrain	B200	32	10.3	330	128	23k	4.6m	14m
MolmoPoint-8B	SFT	B200	80	86.4	6.9k	160	22k	13m	49m
MolmoPoint-8B	SFT-LC	B200	80	36.2	2.9k	160	2k	2.2m	7.5m
MolmoPoint-GUI-8B	SFT	B200	32	9.4	300	128	2k	360k	11m
MolmoPoint-Vid-8B	Pretrain	H100	32	10.1	323	128	23k	3.5m	6.6m
MolmoPoint-Vid-8B	SFT	B200	32	13.7	437	64	6k	1.4m	5.1m
MolmoPoint-Vid-8B	SFT-LC	B200	32	12.5	400	64	800	170k	270k

and the Automatic Mixed Precision (AMP) module[1] for mixed precision training. We do not use sequence-parallelism since we found that a long-context model can fit on the B200 GPUs with just FSDP.

8 VideoPoint Human Evaluation

We perform an internal human preference evaluation with two annotators (two of the authors) on the video pointing outputs of two models – MolmoPoint-8B and the text coordinate baseline. We first manually design the test set by selecting 271 challenging video-query pairs from Molmo2-VideoCountEval [19] and writing 199 new queries for the videos in Molmo2-VideoCaptionEval, resulting in a total of 470 examples in this human preference evaluation dataset. We develop a simple annotation interface to collect pair-wise preferences on this evaluation dataset following standard practice [17], where the ordering of models a and b is randomized, and four choices are allowed: model a is better; model b is better; both are good; or both are bad. After collecting all preferences, we then compute the win rate of MolmoPoint-8B against the baseline, excluding ties (i.e., both good or both bad).

9 Multi-Task Results

We present the full set of video results in Table 12. Comparing Molmo2 and MolmoPoint-8B on video QA, we see mixed outcomes on the long-video QA benchmarks (MLVU (+2.2), Video-MME-SUB (+1.1), VideoEvalPro (-2.4)) and on short-video temporal understanding benchmarks (TempCompass (+1.0), Tomato (-2.3)), with results being about the same elsewhere. We do not see a clear pattern in these results and conclude that the models perform very similarly.

The full set of image benchmarks is shown in Table 13. Compared to Molmo2, we see notable gains on some high-res/OCR-tasks (InfoQa (+2.9), Ai2D (+0.8), ChartQA (+0.8)) and a slight drop in counting benchmarks (CountBench (-0.8), PixMoCount (-0.9)). The consistent gain on high-res/OCR tasks suggests that grounding tokens improved transfer from the pointing training datasets to these tasks, which continues the theme of improving high-res OCR-heavy tasks that we observe with GUI pointing. We attribute the slight drop in counting to the occasional off-by-one errors we observe when counting high-frequency objects (see Figure 6 in the main paper).

There is a significant drop in video captioning performance. However, when we check image-captioning performance, we see the opposite: 54.47 for MolmoPoint-8B vs 53.62 for Molmo2, using the F1 captioning metric from [22]. This again suggests that the way grounding tokens affect video and image performance differs.

[1] https://docs.pytorch.org/docs/stable/amp.html

Table 12 Video benchmark results for a range of proprietary APIs, open-weight baselines, fully-open baselines, Molmo2, and MolmoPoint-8B across video understanding, captioning, and counting benchmarks. The best-performing open-weight model is in **bold**, and the second-best is underlined. For MolmoPoint-8B, we report PerceptionTest val since the test server is no longer operational.

Model	NextQA test [79]	PerceptionTest test [57]	MVBench test [38]	Tomato test [62]	MotionBench val [31]	TempCompass test MCQ [43]	Video-MME test [25]	Video-MME-Sub test [25]	LongVideoBench val [75]	MLVU test MCQ [101]	LVBench test [73]	VideoEvalPro test [49]	Ego Schema test [50]	Molmo2 Caption test F1 Score	Molmo2 Count val accuracy	Short QA avg.	Long QA avg.	Average
API call only																		
GPT-5 [56]	86.3	79.4	74.1	53.0	65.4	80.4	83.3	86.9	72.6	77.7	65.2	68.8	75.6	50.1	35.8	73.1	76.3	70.6
GPT-5 mini [56]	83.2	72.0	66.5	44.1	59.9	74.9	77.3	82.3	69.7	69.1	54.7	60.1	70.9	56.6	29.8	66.8	69.8	65.0
Gemini 3 Pro [28]	84.3	77.6	70.4	48.3	62.6	82.8	88.6	87.5	75.9	75.7	77.0	78.0	68.9	36.0	37.1	71.0	78.8	70.0
Gemini 2.5 Pro [20]	85.3	78.4	70.6	48.6	62.0	81.9	87.8	87.8	76.8	81.5	75.7	78.4	72.2	42.1	35.8	71.1	80.4	71.2
Gemini 2.5 Flash [20]	81.8	74.7	67.0	39.1	59.3	80.2	84.2	84.2	73.1	75.1	64.9	69.6	70.2	46.0	31.9	67.0	74.5	66.7
Claude Sonnet 4.5 [4]	79.2	64.3	62.1	39.6	58.5	72.8	74.2	80.5	65.1	64.0	50.5	50.5	73.1	26.0	27.2	62.8	66.4	59.6
Open weights																		
InternVL3.5-4B [72]	80.3	68.1	71.2	26.8	56.5	68.8	65.4	68.6	60.8	52.0	43.2	46.5	58.9	7.7	26.3	62.0	56.5	53.4
InternVL3.5-8B [72]	81.7	72.7	72.1	24.6	56.6	70.3	66.0	68.6	62.1	53.2	43.4	48.1	58.6	7.8	26.1	63.0	57.1	54.1
Qwen3-VL-4B [84]	81.4	70.7	68.9	31.8	58.6	70.8	69.3	74.0	62.8	58.4	56.2	49.8	68.4	25.2	25.3	63.7	62.7	58.1
Qwen3-VL-8B [84]	83.4	72.7	68.7	35.7	56.9	74.3	71.4	75.2	62.4	57.6	**58.0**	50.3	69.8	26.7	29.6	65.3	63.5	59.5
Keye-VL-1.5-8B [85]	75.8	64.2	56.9	33.0	55.1	**75.5**	**73.0**	**76.2**	66.0	53.8	42.8	54.9	56.3	25.4	27.2	60.1	60.4	55.7
GLM-4.1V-9B [68]	81.3	74.2	68.4	30.0	59.0	72.3	68.2	75.6	65.7	56.6	44.0	51.1	62.6	18.4	26.6	64.2	60.5	56.9
MiniCPM-V-4.5-8B [91]	78.8	70.9	60.5	29.8	59.7	72.7	67.9	73.5	63.9	60.6	50.4	54.9	49.6	29.3	26.3	62.1	60.1	56.6
Eagle2.5-8B [13]	85.0	81.0	74.8	31.0	55.7	74.4	72.4	75.7	66.4	60.4	50.9	58.6	**72.2**	22.8	28.9	67.0	**65.2**	60.7
Fully open																		
PLM-3B [18]	83.4	79.3	74.7	30.9	60.4	69.3	54.9	59.4	57.9	48.4	40.4	46.2	66.9	12.3	24.4	66.3	53.5	53.9
PLM-8B [18]	84.1	**82.7**	**77.1**	33.2	61.4	72.7	58.3	65.4	56.9	52.6	44.5	47.2	68.8	10.9	26.6	68.5	56.2	56.2
LLaVA-Video-7B [97]	83.2	68.8	58.6	24.9	54.2	66.6	63.3	69.7	58.2	52.8	44.2	47.8	57.3	19.9	21.4	59.4	56.2	52.7
VideoChat-Flash-7B [40]	85.5	76.5	74.0	32.5	60.6	69.4	65.3	69.7	64.7	56.0	48.2	51.2	51.3	14.8	21.6	66.4	58.1	56.1
Molmo2-4B	85.5	81.3	75.1	**39.8**	61.6	72.8	69.6	75.7	**68.0**	**63.0**	53.9	59.9	61.2	39.9	34.3	69.3	64.5	62.8
Molmo2-8B	**86.2**	82.1	75.9	39.6	**62.2**	73.4	69.9	75.8	67.5	60.2	52.8	**60.4**	62.0	**43.2**	**35.5**	**69.9**	64.1	**63.1**
Molmo2-O-7B	84.3	79.6	74.8	36.2	60.6	73.0	64.9	69.2	63.7	55.2	49.6	55.1	56.8	40.1	33.2	68.1	59.2	59.7
MolmoPoint																		
MolmoPoint-8B	86.3	82.3*	75.2	37.3	61.4	74.4	69.6	76.9	67.3	62.4	52.5	58.0	63.0	40.3	35.6	69.5	64.2	63.5

10 MolmoPoint-GUISyn Details

We show the data generation pipeline of MolmoPoint-GUISyn in Figure 3. The input to the pipeline is a natural language query, e.g., "a screenshot of AutoCAD", which will be paired with a randomly selected persona from PersonaHub [27] (e.g., *a Sci-fi novelist*)to diversify its content and style.

We systematically construct a comprehensive list of queries by considering screenshot types (desktop, web, mobile), task domains (different websites, Apps, software), platforms (Windows, macOS, iOS, Android, etc.), aspect ratios (4:3, 16:9, etc.), resolutions (720p, 1080p, 4K, etc.), and stages during a task (early, middle, end). We randomly sample and combine those fields to construct inputs that span a broad range of scenarios in the digital world.

We feed the query into our prompt template, and an LLM outputs the corresponding HTML code to render the screenshot. We run customized JavaScript on the HTML code to extract the bounding boxes for all visible elements in the screenshot. Each bounding box contains the synthetic label from its naming attributes in HTML, the corresponding lines of code for this element, the (x, y) center, and the (width, height) of the box.

Table 13 Image benchmark results for a range of proprietary APIs, open-weight baselines, and MolmoPoint-8B across a range of image understanding and counting benchmarks. The result of the best-performing open-weight model is in **bold**, and the second best is underlined.

Model	AI2D test [33]	ChartQA test [51]	DocVQA test [52]	InfoQA test [53]	TextVQA val [63]	VQA v2.0 val [30]	RWQA [78]	MMMU val [95]	MathVista testmini [47]	CountBench [8]	PixMoCount test [22]	MuirBench [70]	MMIU [54]	Blink val [26]	Img QA avg.	Multiimg QA avg.	Average
API call only																	
GPT-5 [56]	89.5	83.8	88.9	83.0	78.7	79.7	80.8	81.8	82.7	90.8	67.2	78.6	71.0	66.5	82.5	72.1	80.2
GPT-5 mini [56]	86.7	82.1	86.7	82.2	79.1	72.1	77.0	78.7	79.2	87.1	74.4	71.4	64.5	68.7	80.5	68.2	77.8
Gemini 2.5 Pro [20]	94.3	77.8	91.5	82.0	70.3	67.1	77.4	79.6	84.6	90.8	73.8	74.5	68.9	73.7	80.8	72.4	79.0
Gemini 2.5 Flash [20]	95.9	76.8	91.1	80.9	73.0	69.4	74.5	79.0	81.2	86.7	63.9	73.5	61.2	70.2	79.3	68.3	76.9
Claude Sonnet 4.5 [4]	91.5	80.2	91.7	65.9	67.2	77.0	61.1	77.8	73.1	87.3	58.3	59.6	54.1	64.8	75.6	59.5	72.1
Open weights only																	
InternVL3.5-4B [72]	82.6	86.0	92.4	78.0	77.9	78.1	66.3	66.6	77.1	82.2	62.4	53.1	49.2	58.1	77.2	53.5	72.1
InternVL3.5-8B [72]	84.0	86.7	92.3	79.1	78.2	79.5	67.5	**73.4**	78.4	79.6	61.9	55.8	49.4	59.5	78.2	54.9	73.2
Qwen3-VL-4B [84]	84.1	85.0	95.3	80.3	81.0	81.7	70.9	67.4	73.7	85.5	58.0	63.8	43.2	65.8	78.4	57.6	74.0
Qwen3-VL-8B [84]	85.7	85.2	**96.1**	**83.1**	82.8	82.3	71.5	69.6	77.2	90.4	65.0	64.4	35.3	**69.1**	80.8	56.3	75.6
Keye-VL-1.5-8B [85]	89.5	85.0	93.4	74.9	81.5	79.3	73.5	71.4	**81.2**	81.6	57.4	51.2	50.3	54.9	79.0	52.1	73.2
GLM-4.1V-9B [68]	87.9	70.0	93.3	80.3	79.6	68.3	70.7	68.0	80.7	88.0	60.7	**74.7**	**62.4**	65.1	77.0	**67.4**	75.0
MiniCPM-V-4.5-8B [91]	86.5	87.4	94.7	73.4	82.2	64.1	72.1	67.7	79.9	83.9	62.8	53.3	46.5	42.0	77.7	47.3	71.2
Eagle2.5-8B [13]	84.5	**87.5**	94.1	80.4	83.7	82.4	76.7	55.8	67.8	90.2	66.9	61.8	48.4	45.8	79.1	52.0	73.3
Fully open																	
PLM-3B [18]	90.9	84.3	93.8	74.6	84.3	84.4	72.4	41.2	59.1	87.1	63.0	25.7	40.6	55.4	75.9	40.6	68.3
PLM-8B [18]	92.7	85.5	94.6	80.0	**86.5**	85.6	75.0	46.1	59.9	91.8	68.0	23.5	27.4	56.0	78.7	35.7	69.5
Molmo2-4B	95.6	86.1	87.8	78.6	85.0	86.6	75.4	50.9	56.7	93.9	88.1	60.5	55.5	57.5	80.4	57.8	75.6
Molmo2-8B	95.8	86.0	93.2	80.1	85.7	87.0	**77.6**	53.0	58.9	93.7	88.5	63.7	54.2	51.3	81.7	56.4	76.3
MolmoPoint-8B-O-7B	93.7	84.9	90.4	77.9	84.7	86.6	73.6	45.8	54.2	**95.1**	**88.9**	58.4	51.7	50.5	79.7	53.5	74.1
MolmoPoint																	
MolmoPoint-8B	**96.4**	86.8	93.8	83.0	86.0	**87.2**	77.4	53.7	59.4	92.9	87.6	62.5	54.6	50.8	**82.2**	56.0	**76.6**

We feed this information back to the LLM to annotate each element with a natural language name (e.g., "Measure Button") and 5 different intents that a real user might ask when interacting with this element.

We use claude-sonnet-4.6 as our coding LLM to generate MolmoPoint-GUISyn, which costs about $0.2 per example, with an average of 54 pointing annotations. Figure 8 demonstrates the qualitative examples from MolmoPoint-GUISyn.

11 MolmoPoint-Track Details

Here, we detail the data generation pipeline for MolmoPoint-Track, which comprises two complementary data sources: (1) MolmoPoint-TrackAny, human-annotated tracks covering a broad range of videos and object categories, and (2) MolmoPoint-TrackSyn, synthetically generated object tracks featuring complex occlusion patterns and motion dynamics.

11.1 MolmoPoint-TrackAny: Human Annotated Tracks

To extend tracking annotations beyond existing datasets, we develop a human-in-the-loop pipeline for annotating object tracks in real videos. Annotating point tracks from scratch is both costly and difficult to quality-control; for multi-object scenes, annotators must identify unique instances, track them simultaneously, and handle highly variable workloads across videos. However, if the objects of interest and their count are known in advance, the task simplifies to tracking a single designated object at a time.

Specifically, we leverage the Molmo2-VideoPoint data, which already provides distinct identities for each text query, and extend single-frame points into full tracks. Figure 4 illustrates the overall data generation pipeline. Each annotation task provides the annotator with the video, the input point for the object of interest, and all

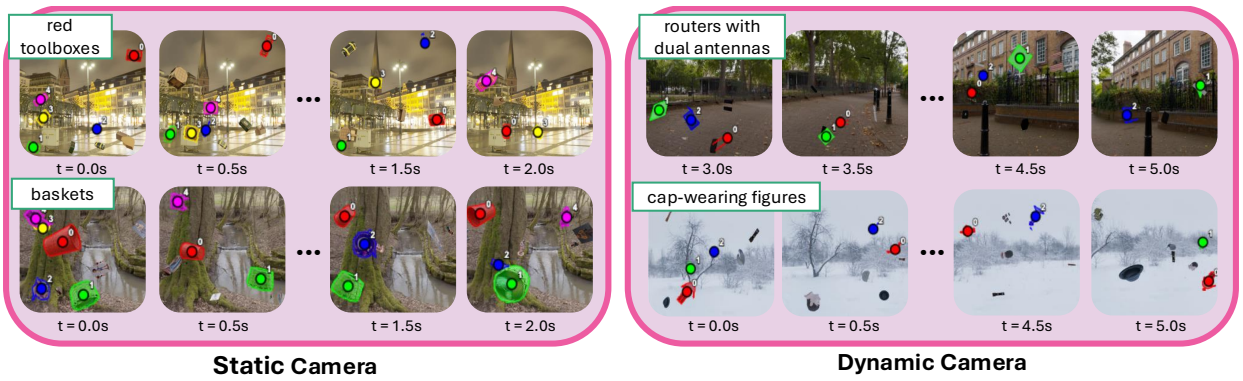


Figure 7 Examples from MolmoPoint-TrackSyn data. Object tracks, text queries, and videos are generated synthetically with static and dynamic camera viewpoints.

other annotated points for context. Annotators track one object at a time while viewing the surrounding points, which helps prevent duplicate tracks for the same instance, particularly in ambiguous cases involving shot changes or visually similar objects. By reducing the cognitive load to single-object tracking, annotators no longer need to jointly identify and track multiple objects. In the end, our annotation consists of 13K videos with 17K text queries and an average of 6.7 unique objects per video, accompanied by diverse tracks with re-identification in shot changes, part-level objects, and complex referring expressions.

11.2 MolmoPoint-TrackSyn: Synthetic Object Tracks and Videos

We generate synthetic multi-object tracking videos in Blender using two pipelines: a static-camera pipeline and a moving-camera pipeline (see Figure 7 for examples). In both cases, each video is rendered with configurable duration, frame rate, resolution, and physically based rendering settings, and the pipeline outputs RGB frames together with per-instance binary masks derived from Blender's object-index pass. The static-camera version uses a fixed camera pose and samples object trajectories relative to a camera-aware frustum, while the moving-camera version extends this setup with a smoothly animated camera and frame-dependent waypoint planning so that visibility constraints are enforced with respect to the camera motion at each waypoint. After generation with diverse motion patterns, the frames are then encoded uniformly into videos as 6 fps.

In both pipelines, 3D assets are sampled from TexVerse [98] after automatic caption-based filtering with GPT. The filter keeps only independently trackable objects and removes scenes, backgrounds, abstract assets, oversized context assets, and collections of unrelated objects, while also assigning noun-only semantic categories to retained assets. For each video, we first build category-specific pools from the filtered assets, randomly choose 1–3 categories, allocate the requested number of objects across these categories, and then sample distinct assets accordingly. This category-based sampling encourages semantic diversity within each sequence while avoiding duplicate object identities.

After selection, assets are imported, merged if needed, normalized to a target scale, centered, and placed on the ground plane. The scene is rendered with randomized lighting and camera parameters. Object motion is generated by sampling waypoint-based trajectories under camera-aware visibility constraints, so objects can be forced to stay visible or move off-screen for selected time spans. The moving-camera pipeline further animates the camera along a smooth trajectory and recomputes visibility constraints with respect to the camera pose over time. For each sequence, we render RGB frames and per-instance segmentation masks, then convert the masks into frame-wise tracking annotations with consistent object identities across time.

After synthesizing the videos, we automatically generate language queries from the selected object captions associated with each sequence. For every video, we collect the source captions of all sampled objects and prompt GPT to produce 0–3 referring queries, where each query must describe a group containing at least two objects. The query generator is constrained to produce concise group-level noun phrases rather than enumerations, avoid explicit counts, and avoid references to object parts, scenes, camera state, or temporal events. It may use either shared fine-grained types (e.g., the toolboxes) or higher-level categories (e.g., the vehicles) when grouping objects. The resulting queries are then matched back to object identities and stored

together with the video path and frame-wise tracking annotations, yielding language-conditioned multi-object tracking examples.

After generating the segmentation masks, we extract the center of the largest connected component to obtain the point tracks. Overall, we have 76k unique queries for 25k videos as our training data, with an average of 3.3 unique objects per video.

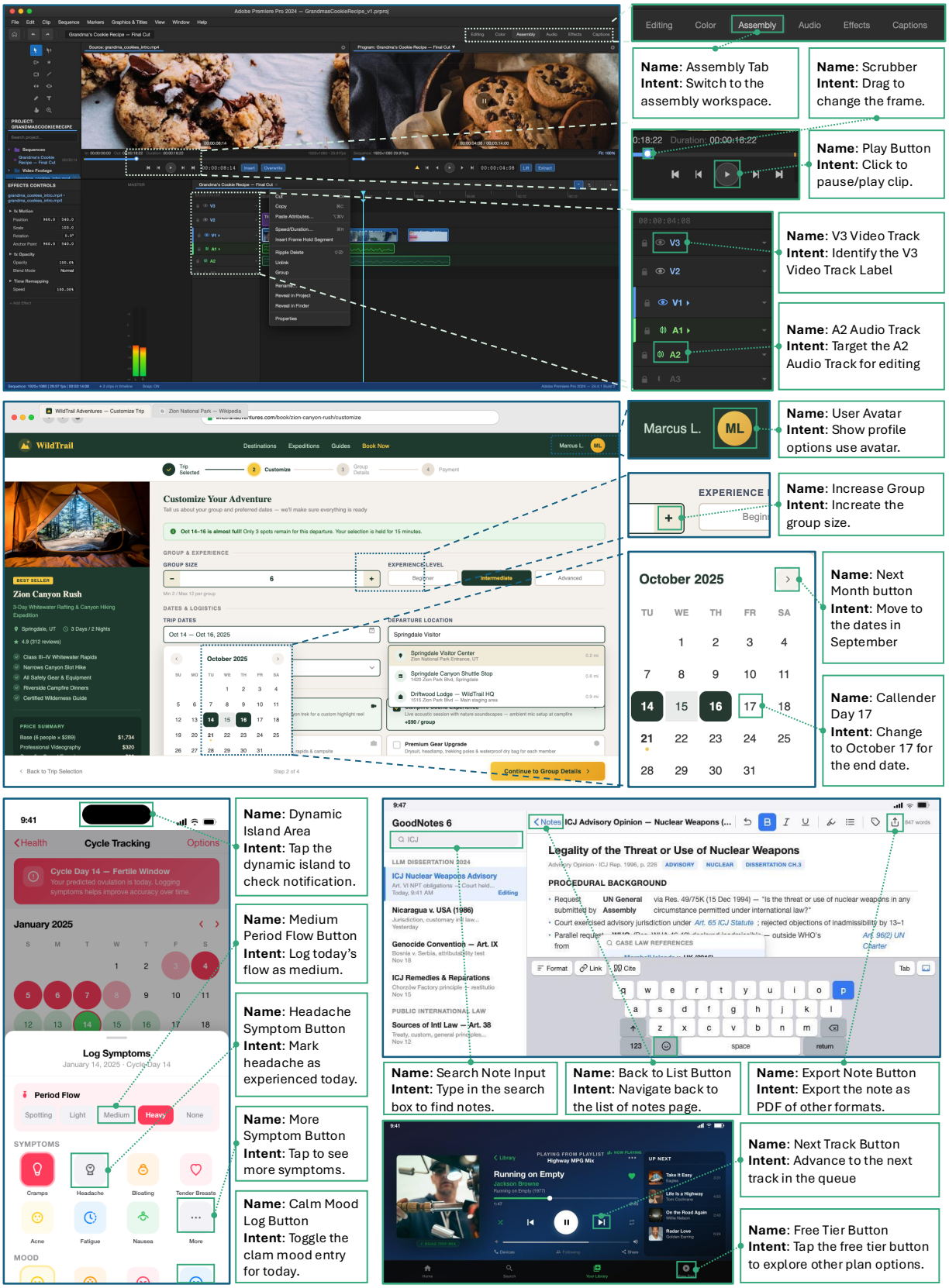


Figure 8 Qualitative examples of MolmoPoint-GUISyn. We demonstrate GUI grounding examples for desktop, web, and mobile screenshots.