Train the Agent, Not the Expert: Learning to Harness Heterogeneous Experts for Multi-Turn Visual Reasoning

Yaowu Fan[1,*] **Tao Han**[2,*] **Dazhao Du**[2] **Andy Jin-Hua Ma**[1,†] **Jia Wan**[3]
[1]Sun Yat-sen University [2]HKUST [3]Harbin Institute of Technology (Shenzhen)
{fywyukee,hantao10200,dudazhao16,jiawan1998}@gmail.com, majh8@mail.sysu.edu.cn

Abstract

Recent advances in computer vision have produced powerful specialized models for detection, segmentation, counting, and other visual tasks. However, these models are typically optimized for isolated tasks, limiting their ability to support general-purpose visual intelligence, particularly when complex language understanding and dense small-object perception are required. We propose **VisHarness**, a trainable visual agent that decouples high-level perception, reasoning, and decision-making from low-level task execution. Rather than training a task-specific model, we train VisHarness to harness a set of carefully designed heterogeneous visual experts. This paradigm preserves the agent's general intelligence while leveraging the precision of specialized visual experts for specific tasks. With only lightweight training, VisHarness learns a generalizable policy for harnessing visual experts, enabling it to solve fundamental vision tasks in diverse complex scenarios through multi-turn interactions with these experts. To enable efficient on-policy reinforcement learning in a live environment, we introduce dynamic visual memory archiving to reduce the visual-token overhead that rapidly accumulates during multi-turn interactions with visual experts. Experiments on four representative benchmarks spanning reasoning segmentation, generalized referring expression segmentation, dense small-object detection, and referring expression counting show that VisHarness substantially outperforms existing general-purpose models and is competitive with or superior to task-specific models. Our code, model weights, and training data are publicly available at https://github.com/fyw1999/VisHarness.

1 Introduction

The past decade has witnessed rapid advances in computer vision. Powerful specialized models have emerged across multiple subfields, such as object detection [33], semantic segmentation [20], and object counting [52], achieving substantial progress on their respective tasks. The continuous improvement of these task-specific capabilities has driven computer vision beyond traditional perception tasks toward more open and complex visual understanding scenarios [7, 24].

Despite their impressive performance, specialized visual models typically operate as iso-

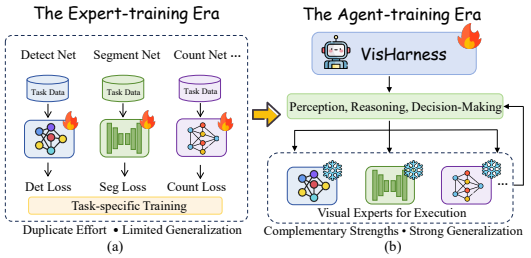

Figure 1: **From expert training to expert harnessing.** (a): Traditional vision methods train a separate specialist for each sub-task. (b): VisHarness learns a harnessing policy over a set of heterogeneous experts and thus can solve complex visual tasks through multi-turn interaction.

*Equal contribution. †Corresponding author. Andy Jin-Hua Ma is also with Guangdong Provincial Key Laboratory of Information Security Technology and Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education, China.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

lated task solvers, each designed for a specific visual task with a fixed input-output format. For example, detection and segmentation models [5, 20, 32, 57] excel at object localization and pixel-level mask generation, but often lack the ability to understand complex instructions. Counting models [11, 39] effectively estimate dense object distributions but provide limited support for instance-level reasoning. Visual grounding models [26, 46] can interpret complex language descriptions, but often struggle with densely packed small objects. As a result, a single model is often insufficient to handle complex visual requests (see Fig. 1(a)). Real-world visual tasks often require combining multiple capabilities, such as first understanding complex expressions, then detecting or segmenting densely packed small objects, and finally verifying the outputs or reasoning based on intermediate results.

In recent years, multimodal large language models (MLLMs) [19, 31] have demonstrated strong vision-language understanding and reasoning capabilities, offering new possibilities for general-purpose visual intelligence. However, MLLMs themselves are not always well suited to precise visual tasks, such as segmentation, dense object detection, and accurate counting. Reliable performance on these tasks often requires task-specific architectural modifications and large amounts of high-quality training data [16, 21, 37, 40]. Recently, some works [10, 14, 17, 56] enable MLLMs to interact with promptable segmentation models [20] over multiple turns to iteratively refine target masks. However, these approaches remain focused on segmentation, leaving other visual tasks that require detecting and counting densely packed small objects largely unexplored.

To address the above issues, we propose **VisHarness**, a trainable high-level agent for harnessing heterogeneous visual experts. With only a small amount of training data, VisHarness can learn a policy for harnessing a carefully designed set of heterogeneous visual expert models, enabling it to solve a range of fundamental vision tasks. The core design principle of VisHarness is to decouple high-level perception, reasoning, and decision-making from low-level vision task execution. Specifically, the agent learns general perception, reasoning, and decision-making abilities, while delegating the execution of specific vision tasks to specialized visual experts. This design preserves the general intelligence of the agent while fully leveraging the precision advantages of specialized visual experts in concrete visual tasks (see Fig. 1(b)). Accordingly, VisHarness can naturally support a variety of fundamental vision tasks, including generalized referring expression segmentation (GRES), reasoning segmentation, referring expression counting, and dense small-objects detection. Through multi-turn interactions with visual experts, VisHarness can not only handle complex visual tasks beyond the capabilities of a single expert but also identify and correct errors from previous turns. At each turn, VisHarness first observes the current image and text content, then reasons to make a decision. After a specific visual expert returns the execution result in visual and textual form, VisHarness can continue reasoning based on this feedback and decide in the next turn whether to refine the current result or invoke another visual expert, until the task is completed.

To learn a generalizable visual expert-harnessing policy, we train VisHarness with on-policy reinforcement learning (RL) in a live environment with deployed visual experts. However, visual feedback accumulates over multiple interaction turns, introducing substantial memory and context overhead that makes RL training difficult. To this end, we further introduce dynamic visual memory archiving, which dynamically archives outdated visual memories at each reasoning step and retains only the visual results produced by the latest interaction with a visual expert. This mechanism preserves necessary historical information while substantially reducing visual context overhead, enabling efficient trajectory sampling for on-policy RL. Since dynamic visual memory archiving changes the historical memory at each turn, we decompose the multi-turn Markov decision process into multiple single-turn Markov decision processes to maintain consistency between training and inference during the stage of RL training. In summary, our main contributions are as follows:

- We introduce VisHarness, a trainable visual agent that decouples high-level visual perception, reasoning, and decision-making from low-level visual task execution. It requires only a small amount of training data to learn a generalizable policy over a set of heterogeneous visual experts and can solve various complex vision tasks and self-correct through multi-turn reasoning.
- We propose a dynamic visual memory archiving mechanism that substantially reduces the visual overhead in multi-turn interactions without compromising historical memory coherence and task performance. This enables efficient on-policy reinforcement learning of VisHarness in a live environment, allowing it to autonomously explore effective and generalizable policies under the guidance of trajectory-level rewards.

- Experimental results on four benchmarks covering generalized referring expression segmentation (GRES), reasoning segmentation, referring expression counting, and dense small-object detection show that VisHarness outperforms general-purpose models and performs comparably to or better than specialized models. Strong performance on tasks and datasets unseen during training further demonstrates the generalizability of the expert harnessing policy learned by VisHarness.

2 Related Work

2.1 MLLM-Based Task-Specific Visual Models

Recent works have adapted MLLMs into specialized visual models [16, 21, 22, 48]. LISA [21] introduces a special <SEG> token and combines its projected hidden representation with image features in a segmentation decoder to generate the target mask for a complex implicit query. Text4Seg++ [22] extends the MLLM vocabulary with 126 semantic brick tokens, each encoding a run of foreground or background cells, enabling the model to autoregressively generate bounding boxes and mask sequences for fine-grained, language-guided segmentation without an additional segmentation decoder. Rex-Omni [16] repurposes the last 1,000 tokens in Qwen2.5-VL's vocabulary as dedicated tokens for quantized relative coordinates, enabling the MLLM to autoregressively generate points, bounding boxes, and polygons to unify open-vocabulary detection, fine-grained referring expression grounding, OCR, and keypoint detection. Despite their effectiveness, these methods usually modify the original MLLM architecture and require training on large-scale datasets for specific downstream tasks, resulting in models specialized for those tasks. In contrast, VisHarness learns a generalizable policy for harnessing visual experts, enabling flexible application across diverse vision tasks.

2.2 Tool-Augmented Visual Agents

Tool-integrated reasoning alleviates the limitations of LLMs in precise computation and access to up-to-date information by invoking external tools such as code interpreters, calculators, and browsers during inference [30, 41]. Beyond text-based tool use, recent works have begun to study tool-integrated agents with visual inputs. OPENTHINKIMG [36] studies tool-augmented visual question answering and uses RL to train MLLMs to adaptively invoke external vision tools. LLaVA-Plus [25] augments multimodal assistants with a repository of vision and vision-language tools for diverse multimodal tasks. Another line of work develops MLLM-based agents that interact with promptable segmentation models [10, 14, 17, 56]. SegAgent [56] emulates human annotation by having an MLLM provide positive and negative point prompts to a promptable segmentation model and iteratively refine segmentation masks through multi-turn interaction. RSAgent [14] supplements a promptable segmentation model with tools such as cropping and rotation, and uses SFT and RL to train an MLLM for multi-turn tool use. However, it still relies on the MLLM to directly generate point prompts for the segmentation model and selects only one target per image–text pair during training. These design settings limit its ability to handle dense multi-object scenes. Unlike these works, VisHarness focuses on learning a generalizable policy to adaptively harness heterogeneous visual experts for diverse fundamental vision tasks under challenging conditions, including scenes with anywhere from zero to hundreds of target objects.

3 VisHarness: Overall Design and Workflow

3.1 Multi-Turn Visual Expert Harnessing

We propose VisHarness, a general-purpose visual agent that decouples high-level visual perception, reasoning, and decision-making from low-level visual task execution. It can effectively solve various complex visual tasks through multi-turn interactions with a set of heterogeneous visual experts. As shown in Fig. 2, given an image-text pair (I_i, T_i), T_i specifies the objects to detect, segment, or count in I_i. The text can be a simple noun phrase (e.g., a class name), a complex attribute description (e.g., relative spatial relations, material, pose, or state), or an implicit query requiring reasoning to identify the target objects. The number of target objects can range from zero to over a hundred.

VisHarness performs multi-turn reasoning over images and text, and finally produces the results $(B_j, M_j)_{j=1}^{\hat{n}}$, where $\hat{n}$ is the number of predicted target objects, and B_j and M_j denote the bounding box and corresponding mask of the j-th object, respectively. At the t-th turn, VisHarness π_θ selects an action $a_t = (\omega_t, \phi_t) \sim \pi_\theta(\cdot \mid \mathcal{M}_t)$ based on its current multimodal memory $\mathcal{M}_t$, where $\omega_t \in \Omega$

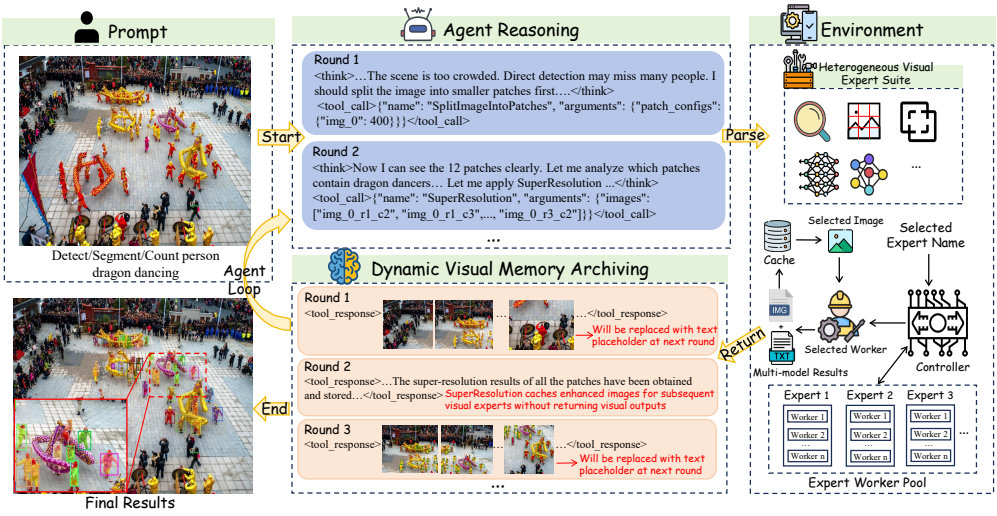

Figure 2: **Overview of VisHarness.** VisHarness solves various complex fundamental vision tasks through multi-turn interactions with heterogeneous visual experts. At each turn, it selects an action based on the current memory. When a visual expert is invoked, the environment parses the expert name and arguments, and a controller dispatches the request to the least-loaded worker for that expert. After receiving the visualized and textual feedback, VisHarness dynamically archives outdated visual memory, incorporates the latest feedback into its memory, and proceeds to the next turn.

denotes the selected visual expert from the expert set Ω, and $\phi_t \in \Phi_{\omega_t}$ denotes its invocation parameters. The environment then parses the action a_t and executes the selected expert ω_t with parameters ϕ_t, producing expert feedback $o_t = (v_t, r_t) \sim \mathcal{E}(\cdot \mid \omega_t, \phi_t)$, where v_t and r_t denote the visual and textual feedback, respectively. Then, the memory is updated by archiving earlier visual feedback and incorporating the current action and its corresponding feedback:

$$\mathcal{M}_{t+1} = \mathcal{U}(\mathcal{A}(\mathcal{M}_t), a_t, o_t), \tag{1}$$

where $\mathcal{A}(\cdot)$ denotes the visual archival function and $\mathcal{U}(\cdot)$ is the memory update function. Specifically, $\mathcal{A}(\cdot)$ replaces earlier visual feedback in the context memory with textual placeholders while preserving the textual history. This reduces the visual context overhead accumulated during multi-turn interactions. This process can be naturally formulated as a Markov decision process (MDP), where $\mathcal{M}_t$ serves as the state, $a_t = (\omega_t, \phi_t)$ serves as the action, and $\mathcal{M}_{t+1}$ is the next state. Note that, to process multiple image-text pairs in parallel, each visual expert model is instantiated as multiple workers in the environment. When VisHarness requests to invoke a specific expert, a controller selects the worker with the lowest current load among all workers of that expert to execute the visual task.

3.2 Heterogeneous Visual Expert Suite

A sufficiently general and well-generalized visual expert suite is an important foundation for VisHarness to handle diverse and complex visual tasks. In our design, each expert performs a key atomic visual task. These experts have their own strengths and limitations, but under the unified coordination of VisHarness, they can complement one another and leverage their respective advantages, thereby solving complex visual tasks that are difficult for any single expert to handle alone.

As shown in Fig. 3, our expert suite consists of six experts, including three visual expert models and three auxiliary operation tools. The visual expert models are responsible for executing specific visual tasks, such as returning object bounding boxes, masks, or center points, while the auxiliary visual tools extend the capability range of these expert models to some extent. For example, in extremely dense scenes with small objects or scenarios with large scale variations, general-purpose visual expert models often struggle to handle the input images directly. In such cases, "SplitImageIntoPatches" divides the entire image into multiple patches, increasing the relative size of small objects within their respective patches. VisHarness can then actively select patches containing the target objects and apply appropriate vision expert models to process them. This overall strategy improves the accuracy of small-object detection and segmentation while enabling different subsequent processing strategies to be selected based on the content of each patch. Furthermore, the "SuperResolution" expert enhances the visual details of objects within a specified patch, potentially helping other vision expert models

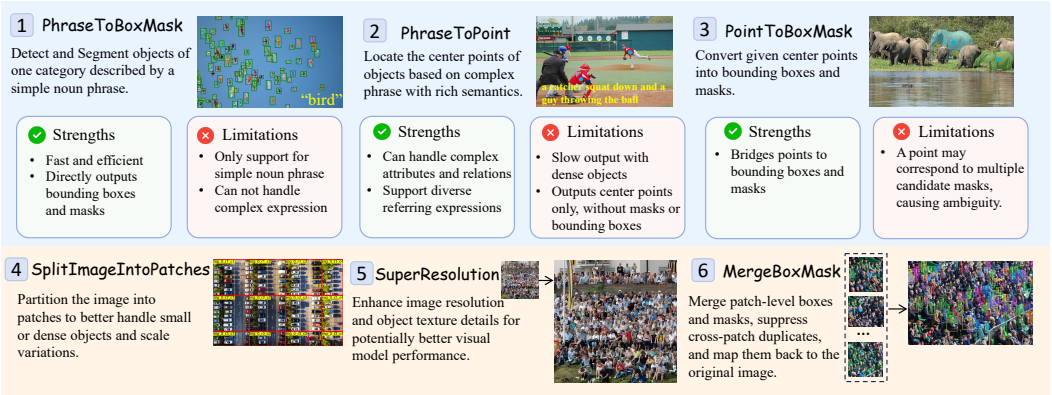

Figure 3: The heterogeneous visual expert suite consists of six visual experts, including three fundamental visual models for executing specific vision tasks and three auxiliary visual tools for extending the capabilities of these models. Each visual expert plays a distinct role, while different experts can complement each other to jointly solve complex visual tasks that are difficult for any single expert to accomplish.

detect and segment blurry objects more accurately. Finally, the "MergeBoxMask" tool leverages the overlapping partitioning strategy of "SplitImageIntoPatches" to remove duplicate detections of the same object across different patches, producing complete and consistent final results.

4 Learning to Harness Visual Experts

VisHarness is trained in two stages. First, we perform supervised fine-tuning (SFT) for cold start, enabling the model to generate visual expert calls in the required format. For on-policy reinforcement learning, we use Group Relative Policy Optimization (GRPO) [8] to train VisHarness in a live system equipped with visual expert models to further enhance its generalization ability. Due to space constraints, please refer to the supplementary material for details on SFT.

4.1 GRPO with Dynamic Visual Memory Archiving

GRPO requires sampling multiple trajectories for each input and optimizes the policy model by comparing their relative advantages within the same group. However, in VisHarness, multi-turn interactions with visual experts cause the returned visualized results to continuously accumulate in an uncontrollable manner, leading to substantial token overhead. To address this issue, we introduce *Dynamic Visual Memory Archiving* during GRPO rollout. The key idea is to preserve only the most recent visual feedback, while dynamically archiving previous visual feedback into compact textual placeholders. In this way, VisHarness can still access the full textual history, while avoiding the continuous accumulation of visual tokens.

Given a training example (I, T, y), where (I, T) is an image-text pair and y denotes the ground-truth visual annotation, we use the old policy model $\pi_{\theta_{\text{old}}}$ to sample a group of G trajectories:

$$\{\tau_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid I, T), \tag{2}$$

where the i-th trajectory is denoted as $\tau_i = \{(\mathcal{M}_{i,t}, a_{i,t}, o_{i,t})\}_{t=1}^{T_i}$. Here, $a_{i,t}$ denotes the action generated by the policy at step t, and $o_{i,t}$ denotes the corresponding observation returned by the selected visual expert model. The action $a_{i,t}$ is sampled conditioned on the current memory $\mathcal{M}_{i,t}$, i.e. $a_{i,t} \sim \pi_{\theta_{\text{old}}}(\cdot \mid \mathcal{M}_{i,t})$. The memory is updated dynamically during rollout using Eq. (1). Concretely, before adding the latest observation, the archiving function $\mathcal{A}$ replaces earlier visual feedback with textual placeholders while preserving the full textual history. The update function $\mathcal{U}$ then incorporates the current action $a_{i,t}$ and its observation $o_{i,t} = (v_{i,t}, r_{i,t})$ into memory. Subsequent actions are thus conditioned on the latest visual feedback together with the textual interaction history.

Since Dynamic Visual Memory Archiving replaces earlier visual feedback with textual placeholders at each turn, the contexts used across turns are not successive prefixes of a single fixed conversation sequence. To ensure context consistency between rollout and training in GRPO, we decompose each complete trajectory into turn-level training samples. Each sample contains the actual contextual memory used at that turn and the corresponding model response, with the training objective applied

only to that response. This enables the model to learn to reason and act based on the corresponding context available at each turn. Specifically, after rolling out G trajectories $\{\tau_i\}_{i=1}^{G}$ of an example (I, T, y), trajectory-level reward $r_i = \mathcal{R}(\hat{y}_i, y)$ is computed for each trajectory τ_i, where $\hat{y}_i$ denotes the predicted visual results and $\mathcal{R}$ is the reward function, whose computation is detailed in the next subsection. Then, the group-relative advantage for the i-th trajectory is computed:

$$\hat{A}_i = \frac{r_i - \text{mean}\left(\{r_j\}_{j=1}^{G}\right)}{\text{std}\left(\{r_j\}_{j=1}^{G}\right) + \epsilon_{\text{adv}}}. \tag{3}$$

Here, $\epsilon_{\text{adv}} > 0$ is a small constant for numerical stability. Since the reward function $\mathcal{R}$ evaluates the overall correctness of each complete trajectory, all turn-level training samples from a trajectory τ_i share its trajectory-level advantage $\hat{A}_i$. Each trajectory τ_i is decomposed into T_i turn-level training samples, with $\hat{A}_i$ assigned to every response token in these samples. For each original image-text pair, the resulting turn-level training dataset $\mathcal{D}_{\text{step}}$ contains a total of $\sum_{i=1}^{G} T_i$ training samples:

$$\mathcal{D}_{\text{step}} = \{x_{i,t} = (\mathcal{M}_{i,t}, a_{i,t}) \mid i = 1, \ldots, G,\ t = 1, \ldots, T_i\}. \tag{4}$$

Let $a_{i,t,k}$ denote the k-th response token at turn t of trajectory τ_i, and let $L_{i,t}$ denote the response length. Finally, the policy model is optimized over the turn-level training samples using the GRPO objective:

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\frac{1}{G}\sum_{i=1}^{G}\frac{1}{T_i}\sum_{t=1}^{T_i}\frac{1}{L_{i,t}}\sum_{k=1}^{L_{i,t}}\Big[\min\Big(\rho_{i,t,k}(\theta)\hat{A}_i, \text{clip}\left(\rho_{i,t,k}(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_i\Big) - \beta D_{\text{KL}}^{i,t,k}(\pi_\theta \| \pi_{\text{ref}})\Big]. \tag{5}$$

where

$$\rho_{i,t,k}(\theta) = \frac{\pi_\theta\big(a_{i,t,k} \mid \mathcal{M}_{i,t}, a_{i,t,<k}\big)}{\pi_{\theta_{\text{old}}}\big(a_{i,t,k} \mid \mathcal{M}_{i,t}, a_{i,t,<k}\big)}, \tag{6}$$

ϵ is the clipping coefficient, β controls the strength of KL regularization, and π_{ref} denotes the fixed reference policy. $D_{\text{KL}}^{i,t,k}$ denotes the KL penalty estimated at response token $a_{i,t,k}$, conditioned on the memory $\mathcal{M}_{i,t}$ and response prefix $a_{i,t,<k}$. By optimizing each action under its corresponding rollout memory using the trajectory-level advantage, this process enables GRPO training for multi-turn visual expert interaction while maintaining context consistency between rollout and training.

4.2 Terminal Visual Rewards

For image-text pairs whose visual ground truth is provided as point annotations, we design a soft reward function based on the relative counting error. For image-text pairs whose visual ground truth is provided as masks, we adopt an IoU-based soft reward function. Specifically, we define the unified soft reward as

$$r = \begin{cases} 8\,\text{IoU}(\hat{z}, z^\star) - 3, & \text{if the visual GT is provided as masks,} \\ \max\left(-3,\ 5 - 8\dfrac{|\hat{n} - n^\star|}{n^\star}\right), & \text{if the visual GT is point annotations,} \end{cases} \tag{7}$$

Here, $z^\star$ and $\hat{z}$ denote the ground-truth and predicted foreground masks, respectively. For multiple targets, their masks are merged by taking the union before computing IoU. $n^\star$ and $\hat{n}$ denote the ground-truth and predicted counts, respectively, with $n^\star > 0$.

5 Experiments

5.1 Evaluation Setup

Benchmarks. We evaluate VisHarness on four benchmarks spanning segmentation, detection, and counting, where the tasks further involve visual grounding conditioned on complex referring phrases, as well as dense- and small-object detection and counting. Specifically, gRefCOCO [24] evaluates generalized referring expression segmentation under single-object, multi-object, and no-object queries. ReasonSeg [21] evaluates reasoning segmentation, requiring the model to infer the target objects from implicit textual queries before segmenting them. Dense200 [16] evaluates dense small-object detection, while REC-8K [7] evaluates referring-expression counting for dense objects described by complex phrases. Dense200 is held out from all SFT and GRPO training data to demonstrate the generalizability of the learned visual expert-harnessing policy. Please refer to the supplementary material for implementation details and evaluation metrics.

Table 1: Generalized Referring Expression Segmentation on gRefCOCO. **Bold**: best.

Method	Training Data Ratio	Validation Set gIoU	Validation Set cIoU	Test Set A gIoU	Test Set A cIoU	Test Set B gIoU	Test Set B cIoU	Avg.
Traditional Specialized Segmentation Models								
LTS [18] (CVPR'2021)	100%	52.70	52.30	62.64	61.87	50.42	49.96	54.98
VLT [9] (ICCV'2021)	100%	52.00	52.51	63.20	62.19	50.88	50.52	55.22
CRIS [47] (CVPR'2022)	100%	56.27	55.34	63.42	63.82	51.79	51.04	56.95
ReLA [24] (CVPR'2023)	100%	63.60	62.42	70.03	69.26	61.02	59.88	64.37
LAVT [53] (CVPR'2022)	100%	58.40	57.64	65.90	65.32	55.83	55.04	59.69
Specialized Segmentation Models Based on MLLM								
LISA(FT) [21] (CVPR'2024)	100%	61.63	61.76	66.27	68.50	58.84	60.63	62.94
GSVA(FT) [50] (CVPR'2024)	100%	**66.47**	**63.29**	**71.08**	69.93	62.23	60.47	**65.58**
LIRA-8B [23] (ICCV'2025)	0%	36.70	40.90	50.40	52.40	42.40	44.90	44.62
SELF1E-8B [54] (CVPR'2026)	0%	37.50	44.40	53.20	57.50	45.60	50.90	48.18
Expert-Harnessing Visual Agents								
Qwen3-VL-8B (w/ experts)	0%	44.61	24.36	25.79	21.45	25.05	16.65	26.32
VisHarness-SFT (only SFT)	0.76%	51.45	57.04	67.40	68.13	60.91	60.64	60.93
VisHarness-RL (only RL)	0.76%	56.37	57.43	69.98	68.86	61.23	57.43	61.88
VisHarness	0.76%	56.92	60.38	70.53	**70.63**	**63.27**	**62.10**	63.97

5.2 Comparison with State-of-the-Art Methods

5.2.1 Generalized Referring Expression Segmentation

The GRES task requires the model to segment specific targets in an image according to a given sentence. The target referred to by the sentence may be absent from the image, or may correspond to one or multiple object instances. Table 1 presents the comparison results between our method and other methods specifically designed for referring expression segmentation. VisHarness achieves an average score of 63.97, only 1.61 and 0.40 points below GSVA(FT) and ReLA, respectively, while achieving the best cIoU on testA and the best gIoU and cIoU on testB. Both GSVA(FT) and ReLA use the full gRefCOCO training set, and GSVA(FT) additionally fine-tunes a pretrained MLLM using ten other segmentation datasets during mixed training [50]. In contrast, although VisHarness also builds on a pretrained MLLM and vision expert models, it achieves results comparable to or even better than those of specialized models using only about 0.76% of the gRefCOCO training set. This indicates that our method can learn a generalizable vision expert harnessing policy from very limited data, substantially reducing reliance on task-specific training data while achieving strong performance.

Compared with Qwen3-VL-8B (w/ tools), VisHarness-SFT raises the average score from 26.32 to 60.93 (+34.61 points), showing that effective task execution requires learning an expert-harnessing policy rather than simply equipping a pretrained MLLM with visual experts. RL further improves all six metrics, raising the average to 63.97 and demonstrating its effectiveness in refining the policy beyond SFT. Without distillation from larger models, VisHarness-RL outperforms VisHarness-SFT overall using only trajectory-level task rewards, further supporting the need for RL training.

5.2.2 Reasoning Segmentation

In reasoning segmentation [21], the target object is not explicitly identified in the text but must instead be inferred from the textual description. VisHarness naturally supports this task through its "perception–reasoning–decision–execution" capability: the MLLM first infers the target object from the text and then passes it as an argument to the visual expert for segmentation. As shown in Table 2, VisHarness achieves the best overall performance, with an average score of 70.2, surpassing SELF1E-8B and SAM3-Agent (Qwen2.5-VL-7B) by 3.1 and 13.2 points, respectively. It also achieves the best results on three of the four evaluation metrics.

Table 2: Comparison with state-of-the-art reasoning segmentation methods on ReasonSeg.

Method	Val gIoU	Val cIoU	Test gIoU	Test cIoU	Avg.
LISA [21] (CVPR'2024)	53.6	52.3	48.7	48.8	50.9
SegLLM [44] (ICLR'2025)	57.2	54.3	52.4	48.4	53.1
Seg-Zero [27] (arXiv'2025)	62.6	62.0	57.5	52.0	58.5
HiMTok-8B [43] (ICCV'2025)	60.7	67.0	60.8	66.2	63.7
Text4Seg++ [22] (TPAMI'2026)	59.1	49.5	57.1	52.1	54.5
SELF1E-8B [54] (CVPR'2026)	65.9	**69.7**	65.7	67.0	67.1
SAM3-Agent [5] (ICLR'2026)	62.2	49.1	63.0	53.5	57.0
Qwen3-VL-8B (w/ experts)	3.6	1.3	4.2	5.2	3.6
VisHarness-SFT (only SFT)	73.0	62.7	73.3	67.1	69.0
VisHarness-RL (only RL)	73.0	62.6	71.6	61.2	67.1
VisHarness	**73.1**	62.1	**73.8**	**71.8**	**70.2**

Compared with Qwen3-VL-8B (w/ tools), VisHarness-SFT raises the average score from 3.6 to 69.0. The baseline frequently omits the required `MergeBoxMask` call for final submission, yielding empty predictions and highlighting the importance of learning a complete expert-harnessing policy through SFT. RL further raises the average to 70.2, with the largest gain of 4.7 points in test cIoU. The modest gain may reflect the task's focus on inferring targets from implicit queries, with a relatively fixed expert-calling strategy and limited room for expert coordination. This may also explain why VisHarness-RL, trained solely with task-outcome rewards, achieves a lower average score than VisHarness-SFT (67.1 vs. 69.0).

5.2.3 Dense and Tiny Object Detection

Table 3 presents the dense small-object detection results on the Dense200 dataset. We compare against the open-set detector GroundingDINO [26], general-purpose MLLMs, and specialist detection models. Notably, our training data do not include any Dense200 samples. Nevertheless, VisHarness-SFT and VisHarness achieve the best and second-best mean F1 scores of 68.7 and 67.5, respectively. VisHarness surpasses Rex-Omni and SenseNova-Vision by 9.2 and 0.7 points in mean F1. These strong results on a detection benchmark outside the training tasks and datasets demonstrate the generalizability of the learned expert-harnessing policy.

However, RL slightly reduces F1@IoU 0.5 from 83.9 to 82.7, F1@IoU 0.95 from 16.8 to 15.8, and mean F1 from 68.7 to 67.5. A likely explanation is the mismatch between the RL objective and detection evaluation. RL uses no detection-task data. For REC-8K, the RL supervision consists of object counts derived from point annotations rather than bounding boxes. The reward in Eq. 7 compares predicted and ground-truth counts without assessing box localization. This misalignment may limit further gains in box quality. It may also explain why VisHarness-RL falls well below VisHarness-SFT (60.0 vs. 68.7 mean F1).

Table 3: Comparison on Dense200. All agents use Qwen3-VL-8B. Unreported results are denoted by –.

Method	Venue	F1@IoU 0.5	F1@IoU 0.95	Mean F1
Traditional Open-Set Detectors				
Grounding DINO [26]	ECCV'2024	36.9	**19.7**	33.1
General-Purpose MLLMs				
DeepSeek-VL2-Tiny [49]	arXiv'2024	2.2	0.3	1.5
OVIS2.5-9B [28]	arXiv'2025	14.0	0.0	5.1
OVIS2.5-2B [28]	arXiv'2025	17.9	0.0	6.7
MiMo-VL [51]	arXiv'2025	29.7	0.4	15.9
Qwen2.5-VL-3B [3]	arXiv'2025	0.8	0.1	0.5
Qwen2.5-VL-7B [3]	arXiv'2025	1.1	0.1	0.6
Qwen3-VL-8B [4]	arXiv'2025	15.90	2.41	11.44
Qwen3.5-9B [31]	–	40.46	7.76	30.02
DeepSeek-VL2-Small [49]	arXiv'2024	16.0	3.9	12.7
SEED1.5-VL [12]	arXiv'2025	76.9	5.3	53.2
Moondream 3.1 [29]	–	74.61	–	–
Specialized Visual Models Based on MLLM				
Rex-Omni-SFT [16]	CVPR'2026	60.2	10.6	46.4
Rex-Omni [16]	CVPR'2026	78.4	10.3	58.3
LocateAnything [42]	ECCV'2026	74.0	<u>18.5</u>	58.7
SenseNova-Vision [13]	arXiv'2026	–	–	66.8
Expert-Harnessing Visual Agents				
Qwen3-VL (w/ experts) [4]	arXiv'2025	10.6	1.6	8.4
VisHarness-SFT (only SFT)	Ours	**83.9**	16.8	**68.7**
VisHarness-RL (only RL)	Ours	73.3	14.8	60.0
VisHarness	Ours	<u>82.7</u>	15.8	<u>67.5</u>

5.2.4 Referring Expression Counting and Localization

Table 4 shows that VisHarness achieves slightly lower counting and localization performance than GDino (FT) and GroundingREC, both fine-tuned on REC-8K. However, it also supports complex language-guided pixel-level segmentation, offering broader applicability. It outperforms Molmo2-4B, the model behind expert `PhraseToPoint`, in test MAE (8.91 vs. 10.04) and F1 (0.59 vs. 0.51), highlighting the benefit of coordinating heterogeneous experts over a single expert. As in other tasks, the untrained agent performs poorly, while RL improves SFT's test MAE from 10.34 to 8.91 and F1 from 0.56 to 0.59. VisHarness-RL has slightly worse counting performance but better localization than VisHarness-SFT.

Table 4: Comparison of specialized methods and general-purpose MLLMs on REC-8K. F1: point localization; FT: fine-tuning on REC-8K.

Method	Val set			Test set		
	MAE↓	RMSE↓	F1↑	MAE↓	RMSE↓	F1↑
Mean	14.28	27.75	–	13.75	25.91	–
ZSC (ResNet-50) [52] (CVPR'2023)	14.84	31.30	–	14.93	29.72	–
ZSC (Swin-T) [52] (CVPR'2023)	12.96	26.74	–	13.00	29.07	–
TFOC [35] (WACV'2024)	16.08	31.61	0.12	17.27	32.68	0.11
CounTX [1] (BMVC'2023)	11.88	27.04	–	11.84	25.62	–
CountGD [2] (NeurIPS'2024)	9.51	22.91	–	11.33	30.87	–
GDino [26] (ECCV'2024)	11.77	28.60	0.34	11.71	26.97	0.35
GDino (FT) [26] (ECCV'2024)	9.03	<u>21.98</u>	<u>0.65</u>	<u>8.88</u>	<u>21.95</u>	<u>0.66</u>
GroundingREC [7] (CVPR'2024)	**6.80**	**18.13**	**0.68**	**6.50**	**19.79**	**0.69**
Qwen3-VL-8B [4] (arXiv'2025)	11.62	29.33	0.40	13.08	36.97	0.38
Qwen3.5-9B [31]	9.78	24.90	0.45	10.18	31.44	0.47
Molmo2-4B [6, 38] (CVPR'2026)	–	–	–	10.04	32.18	0.51
Qwen3-VL-8B (w/ experts)	16.15	31.39	0.13	16.73	35.04	0.13
VisHarness-SFT (only SFT)	9.85	23.18	0.56	10.34	26.52	0.56
VisHarness-RL (only RL)	10.00	29.67	0.58	10.52	33.22	0.58
VisHarness	<u>8.67</u>	24.76	0.59	8.91	25.24	0.59

5.3 In-Depth Analysis

5.3.1 Effect of Multi-Turn Interaction

Fig. 4 presents interaction-turn ablations on GRES and referring expression counting. We vary the maximum number of interactions between the MLLM and visual experts from 2 to 12, reporting gIoU and cIoU for GRES, MAE and RMSE for counting, and completion rates for both tasks. Low turn budgets yield poor task performance and low completion rates, highlighting the importance of multi-turn interaction. GRES performance and completion rate largely stabilize after approximately six turns, whereas referring expression counting requires approximately eight interaction turns for its performance and completion rate to stabilize.

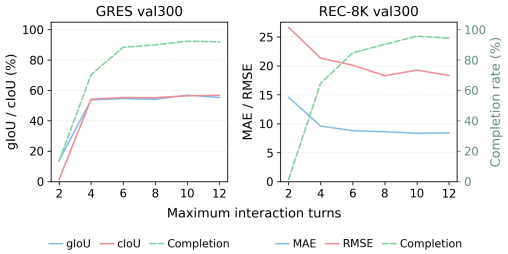

Figure 4: Effect of the number of interaction turns on performance and completion rate.

Table 5: Effect of the Dynamic Visual Memory Archiving. Tokens denotes the average number of visual tokens per trajectory, and KV denotes peak active KV-cache memory per GPU (GiB).

GRES	gIoU ↑	cIoU ↑	Tokens ↓	KV ↓
w/o	**56.24**	55.76	4,608.37	7.66
w/	55.82	**56.32**	**1,080.70**	**5.05**
REC-8K	MAE ↓	RMSE ↓	Tokens ↓	KV ↓
w/o	7.97	16.27	15,253.19	22.03
w/	**7.84**	**15.81**	**4,136.22**	**10.16**

5.3.2 Effect of Dynamic Visual Memory Archiving

Table 5 compares VisHarness with and without Dynamic Visual Memory Archiving on subsets of the GRES and referring expression counting validation sets. Both settings achieve comparable task performance, suggesting that replacing outdated visual feedback from experts with textual placeholders reduces visual tokens while retaining sufficient information for subsequent decision-making. Specifically, archiving reduces average visual tokens per trajectory by 76.5% and 72.9%, respectively, substantially lowering the MLLM's context overhead and making both SFT and RL training more memory efficient. After training, we deploy VisHarness with vLLM for inference. Archiving also reduces peak active KV-cache memory per GPU by 34.1% and 53.9%, respectively.

5.3.3 Visual Tools Usage Distribution Analysis

Fig. 5 visualizes the distributions of visual expert invocations across different models and tasks. Fig. 5(a) and Fig. 5(b) compare Qwen3-VL-8B and VisHarness on Dense200. VisHarness increases the `MergeBoxMask` call ratio from 8.7% to 33.7%. Since `MergeBoxMask` is required to submit the final predictions, the baseline's frequent omission of this call helps explain its poor performance across tasks. Fig. 5(b) and Fig. 5(d) compare VisHarness on Dense200 and gRefCOCO. On Dense200, VisHarness primarily uses `PhraseToBoxMask` for detection based on class names. On gRefCOCO, it relies more on `PhraseToPoint` to localize targets described by complex referring expressions before segmentation. This indicates that VisHarness adapts its expert invocation pattern to task requirements. Fig. 5(c) and Fig. 5(d) compare VisHarness-SFT and VisHarness on gRefCOCO. RL reduces the `PhraseToBoxMask` call ratio from 13.0% to 8.6% and increases that of `PhraseToPoint` from 37.1% to 39.3%, further reinforcing the expert invocation pattern needed for this task.

5.3.4 Qualitative Analysis

Fig. 6 shows two representative examples of multi-turn interaction. For GRES, VisHarness simultaneously locates the pink tie and pink phone with `PhraseToPoint`, then converts their centers into masks with `PointToBoxMask`. An unsuitable `mode` selects incorrect masks among multiple candidates for each point. VisHarness detects the error from visual feedback and switches `mode` from `area` to `confidence`, recovering the correct masks and demonstrating active error detection and correction. For referring expression counting, the query requests all people wearing black coats. VisHarness uses `SplitImageIntoPatches` to increase the relative size of these small, dense targets and `SuperResolution` to enhance texture details. It selects only patches containing targets

for localization with `PhraseToPoint`, converts the points into masks and bounding boxes with `PointToBoxMask`, and removes duplicates across patches with `MergeBoxMask`. This demonstrates how partitioning simplifies dense small-object tasks while selective patch processing focuses on relevant regions. More visualizations and analyses are provided in Appendix B.

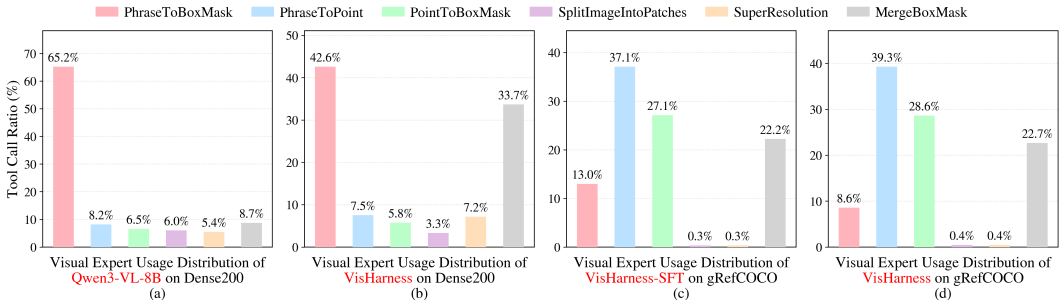

Figure 5: Distribution of visual expert calling by different model variants across different datasets.



Figure 6: Multi-turn interaction visualization on two representative image-text pairs.

6 Conclusion

In this paper, we presented **VisHarness**, a trainable visual agent that learns to harness heterogeneous visual experts for complex fundamental vision tasks. Instead of adapting an MLLM to solve a single visual task, VisHarness decouples high-level perception, reasoning, and decision-making from low-level visual execution, allowing the agent to preserve general vision-language reasoning ability while leveraging the precision of specialized visual models. Through multi-turn interactions with visual experts, VisHarness can adaptively select and coordinate heterogeneous visual experts at each turn, and self-correct to solve visual tasks involving complex language understanding and dense small-object perception. The dynamic visual memory archiving mechanism enables efficient reinforcement learning in a live environment with deployed visual experts without compromising contextual memory coherence or final task performance, further improving the generalizability of learned policy. Quantitative experiments show that VisHarness achieves comparable or superior performance to specialized models across tasks. Qualitative analysis further demonstrates that online reinforcement learning refines the distribution of expert invocations to better align with task-specific preferences, thereby improving task performance.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China (Grant Nos. 62276281 and U22A2095) and the Guangdong Basic and Applied Basic Research Foundation, China (Grant No. 2024A1515011882). The authors declare no competing interests.

References

[1] Niki Amini-Naieni, Kiana Amini-Naieni, Tengda Han, and Andrew Zisserman. Open-world text-specified object counting. In *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press, 2023.

[2] Niki Amini-Naieni, Tengda Han, and Andrew Zisserman. Countgd: Multi-modal open-world counting. *Advances in Neural Information Processing Systems*, 37:48810–48837, 2024.

[3] Shuai Bai, Keqin Chen, Xuejing Liu, et al. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[4] Shuai Bai et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[5] Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. Sam 3: Segment anything with concepts. In *The Fourteenth International Conference on Learning Representations*, 2026.

[6] Christopher Clark, Jieyu Zhang, Zixian Ma, Jae Sung Park, Rohun Tripathi, Sangho Lee, Mohammadreza Salehi, Jason Ren, Chris Dongjoo Kim, Yinuo Yang, Vincent Shao, Yue Yang, Weikai Huang, Ziqi Gao, Taira Anderson, Jianrui Zhang, Jitesh Jain, George Stoica, Ali Farhadi, and Ranjay Krishna. Molmo2: Open weights and data for vision-language models with video understanding and grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 28652–28668, June 2026.

[7] Siyang Dai, Jun Liu, and Ngai-Man Cheung. Referring expression counting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16985–16995. IEEE, 2024.

[8] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[9] Henghui Ding, Chang Liu, Suchen Wang, and Xudong Jiang. Vision-language transformer and query generation for referring segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16321–16330, 2021.

[10] Tianyuan Du, Haopeng Li, Zhen Fan, Jiarui Zhang, Panwang Pan, and Yang Zhang. Sam-veteran: An mllm-based human-like sam agent for reasoning segmentation. In *International Conference on Learning Representations*, volume 2026, pages 145413–145436, 2026.

[11] Yaowu Fan, Jia Wan, Tao Han, Antoni B. Chan, and Andy J. Ma. Video individual counting for moving drones. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12284–12293, 2025.

[12] Dong Guo, Faming Wu, Feida Zhu, et al. Seed1.5-VL technical report. *arXiv preprint arXiv:2505.07062*, 2025.

[13] Xiaoyang Han, Jianhua Li, Kewang Deng, Zukai Chen, Xuanke Shi, Sihan Wang, Boxuan Li, Linyan Wang, Siyi Xie, Xin You, Jinsheng Quan, Zhongang Cai, Haiwen Diao, Ziwei Liu, Lei Yang, Dahua Lin, and Quan Wang. Vision as unified multimodal generation. *arXiv preprint arXiv:2607.06560*, 2026.

[14] Xingqi He, Yujie Zhang, Shuyong Gao, Wenjie Li, Lingyi Hong, Mingxi Chen, Kaixun Jiang, Jiyuan Fu, and Wenqiang Zhang. RSAgent: Learning to reason and act via multi-turn tool invocations for text-guided segmentation. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 42242–42266, 2026.

[15] Jack Hong, Chenxiao Zhao, ChengLin Zhu, Weiheng Lu, Guohai Xu, and Xing Yu. Deepeyesv2: Toward agentic multimodal model. *arXiv preprint arXiv:2511.05271*, 2025.

[16] Qing Jiang, Junan Huo, Xingyu Chen, Yuda Xiong, Zhaoyang Zeng, Yihao Chen, Tianhe Ren, Junzhi Yu, and Lei Zhang. Detect anything via next point prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

[17] Yankai Jiang, Qiaoru Li, Binlu Xu, Haoran Sun, Chao Ding, Junting Dong, Yuxiang Cai, Xuhong Zhang, and Jianwei Yin. Ibisagent: Reinforcing pixel-level visual reasoning in mllms for universal biomedical object referring and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20996–21005, June 2026.

[18] Ya Jing, Tao Kong, Wei Wang, Liang Wang, Lei Li, and Tieniu Tan. Locate then segment: A strong pipeline for referring image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9858–9867, 2021.

[19] Kimi Team. Kimi K2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

[20] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollar, and Ross Girshick. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4015–4026, October 2023.

[21] Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Jiaya 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.

[22] Mengcheng Lan, Chaofeng Chen, Jiaxing Xu, Zongrui Li, Yiping Ke, Xudong Jiang, Yingchen Yu, Yunqing Zhao, and Song Bai. Text4seg++: Advancing image segmentation via generative language modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(8): 9486–9501, 2026.

[23] Zhang Li, Biao Yang, Qiang Liu, Shuo Zhang, Zhiyin Ma, Liang Yin, Linger Deng, Yabo Sun, Yuliang Liu, and Xiang Bai. Lira: Inferring segmentation in large multi-modal models with local interleaved region assistance. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 24056–24067, 2025.

[24] Chang Liu, Henghui Ding, and Xudong Jiang. Gres: Generalized referring expression segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23592–23601. IEEE, 2023.

[25] Shilong Liu, Hao Cheng, Haotian Liu, Hao Zhang, Feng Li, Tianhe Ren, Xueyan Zou, Jianwei Yang, Hang Su, Jun Zhu, et al. Llava-plus: Learning to use tools for creating multimodal agents. In *European conference on computer vision*, pages 126–142. Springer, 2024.

[26] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European conference on computer vision*, pages 38–55. Springer, 2024.

[27] Yuqi Liu, Bohao Peng, Zhisheng Zhong, Zihao Yue, Fanbin Lu, Bei Yu, and Jiaya Jia. Seg-zero: Reasoning-chain guided segmentation via cognitive reinforcement. *arXiv preprint arXiv:2503.06520*, 2025.

[28] Shiyin Lu, Yang Li, Yu Xia, et al. Ovis2.5 technical report. *arXiv preprint arXiv:2508.11737*, 2025.

[29] Moondream. Moondream 3.1: Beyond benchmarks. `https://moondream.ai/blog/moondream-3-1-beyond-benchmarks`, July 2026. Official model release blog.

[30] Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*, 2025.

[31] Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026.

[32] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[33] Ranjan Sapkota, Rahul Harsha Cheppally, Ajay Sharda, and Manoj Karkee. Yolo26: key architectural enhancements and performance benchmarking for real-time object detection. *arXiv preprint arXiv:2509.25164*, 2025.

[34] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, page 1279–1297. Association for Computing Machinery, 2025.

[35] Zenglin Shi, Ying Sun, and Mengmi Zhang. Training-free object counting with prompts. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 323–331, 2024.

[36] Zhaochen Su, Linjie Li, Mingyang Song, Yunzhuo Hao, Zhengyuan Yang, Jun Zhang, Guanjie Chen, Jiawei Gu, Juntao Li, Xiaoye Qu, et al. Openthinkimg: Learning to think with images via visual tool reinforcement learning. *arXiv preprint arXiv:2505.08617*, 2025.

[37] Hao Tang, Chenwei Xie, Haiyang Wang, Xiaoyi Bao, Tingyu Weng, Pandeng Li, Yun Zheng, and Liwei Wang. Ufo: A unified approach to fine-grained visual perception via open-ended language interface. *arXiv preprint arXiv:2503.01342*, 2025.

[38] Kostas Triaridis, Panagiotis Kaliosis, E-Ro Nguyen, Jingyi Xu, Dimitris Samaras, and Hieu Le. What is the right embedding space for contrastive learning in referring expression counting? In *European Conference on Computer Vision (ECCV)*, 2026.

[39] Jia Wan, Ziquan Liu, and Antoni B Chan. A generalized loss function for crowd counting and localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1974–1983, 2021.

[40] Haiyang Wang, Hao Tang, Li Jiang, Shaoshuai Shi, Muhammad Ferjad Naeem, Hongsheng Li, Bernt Schiele, and Liwei Wang. Git: Towards generalist vision transformer through universal language interface. In *Computer Vision – ECCV 2024*, pages 55–73. Springer Nature Switzerland, 2025.

[41] Hongru Wang, Cheng Qian, Wanjun Zhong, Xiusi Chen, Jiahao Qiu, Shijue Huang, Bowen Jin, Mengdi Wang, Kam-Fai Wong, and Heng Ji. Acting less is reasoning more! teaching model to act efficiently. *arXiv preprint arXiv:2504.14870*, 2025.

[42] Shihao Wang, Shilong Liu, Yuanguo Kuang, Xinyu Wei, Yangzhou Liu, Zhiqi Li, Yunze Man, Guo Chen, Andrew Tao, Guilin Liu, Jan Kautz, Lei Zhang, and Zhiding Yu. LocateAnything: Fast and high-quality vision-language grounding with parallel box decoding. In *Computer Vision – ECCV 2026*, volume 17066 of *Lecture Notes in Computer Science*, pages 336–357. Springer, 2026.

[43] Tao Wang, Changxu Cheng, Lingfeng Wang, Senda Chen, and Wuyue Zhao. Himtok: Learning hierarchical mask tokens for image segmentation with large multimodal model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 23267–23278, October 2025.

[44] X. Wang, S. Zhang, S. Li, K. Li, K. Kallidromitis, Y. Kato, K. Kozuka, and T. Darrell. Segllm: Multi-round reasoning segmentation with large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[45] Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. Real-esrgan: Training real-world blind super-resolution with pure synthetic data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 1905–1914, October 2021.

[46] Yabing Wang, Zhuotao Tian, Zheng Qin, Sanping Zhou, and Le Wang. Refdetector: A simple yet effective matching-based method for referring expression comprehension. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 8033–8041, 2025.

[47] Zhaoqing Wang, Yu Lu, Qiang Li, Xunqiang Tao, Yandong Guo, Mingming Gong, and Tongliang Liu. Cris: Clip-driven referring image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11686–11695, 2022.

[48] Cong Wei, Yujie Zhong, Haoxian Tan, Yingsen Zeng, Yong Liu, Hongfa Wang, and Yujiu Yang. Instructseg: Unifying instructed visual segmentation with multi-modal large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 20193–20203, 2025.

[49] Zhiyu Wu, Xiaokang Chen, Zizheng Pan, et al. DeepSeek-VL2: Mixture-of-experts vision-language models for advanced multimodal understanding. *arXiv preprint arXiv:2412.10302*, 2024.

[50] Zhuofan Xia, Dongchen Han, Yizeng Han, Xuran Pan, Shiji Song, and Gao Huang. Gsva: Generalized segmentation via multimodal large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3858–3869, 2024.

[51] Xiaomi LLM-Core Team, Zihao Yue, Zhenru Lin, Yifan Song, et al. MiMo-VL technical report. *arXiv preprint arXiv:2506.03569*, 2025.

[52] Jingyi Xu, Hieu Le, Vu Nguyen, Viresh Ranjan, and Dimitris Samaras. Zero-shot object counting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15548–15557, June 2023.

[53] Zhao Yang, Jiaqi Wang, Yansong Tang, Kai Chen, Hengshuang Zhao, and Philip H. S. Torr. LAVT: Language-aware vision transformer for referring image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18155–18165, 2022.

[54] Anqi Zhang, Xiaokang Ji, Guangyu Gao, Jianbo Jiao, Chi Harold Liu, and Yunchao Wei. Rethinking mllm itself as a segmenter with a single segmentation token. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19196–19207, June 2026.

[55] Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, Wenmeng Zhou, and Yingda Chen. SWIFT: A scalable lightWeight infrastructure for fine-tuning. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence*, volume 39, pages 29733–29735, 2025.

[56] Muzhi Zhu, Yuzhuo Tian, Hao Chen, Chunluan Zhou, Qingpei Guo, Yang Liu, Ming Yang, and Chunhua Shen. Segagent: Exploring pixel understanding capabilities in mllms by imitating human annotator trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3686–3696, June 2025.

[57] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

A Additional Implementation Details

A.1 Evaluation Metrics

For segmentation tasks, we report gIoU and cIoU. Given the predicted mask $\hat{M}_i$ and the ground-truth mask M_i for the i-th sample, gIoU is computed as the average IoU over all N evaluation samples:

$$\mathrm{gIoU} = \frac{1}{N}\sum_{i=1}^{N}\frac{|\hat{M}_i \cap M_i|}{|\hat{M}_i \cup M_i|}. \tag{8}$$

For samples with no target objects, the per-sample IoU is set to 1 only when the model explicitly returns an empty final prediction, and to 0 otherwise. Multiple object masks are combined into a single foreground mask, and ignored pixels are excluded from evaluation. cIoU measures the cumulative intersection-over-union over the whole dataset:

$$\mathrm{cIoU} = \frac{\sum_{i=1}^{N}|\hat{M}_i \cap M_i|}{\sum_{i=1}^{N}|\hat{M}_i \cup M_i|}. \tag{9}$$

If the cumulative union is zero, cIoU is defined as 0.

For detection on Dense200, we report F1 scores at IoU thresholds of 0.5 and 0.95, denoted as $\mathrm{F1}_{0.5}$ and $\mathrm{F1}_{0.95}$, together with mF1. For each threshold τ, we process ground-truth boxes in order, matching each to the unused prediction with the highest IoU if this IoU is at least τ. Precision and recall are computed for each image-query pair and averaged separately over all N samples:

$$P_\tau = \frac{1}{N}\sum_{i=1}^{N}\frac{\mathrm{TP}_{i,\tau}}{\mathrm{TP}_{i,\tau}+\mathrm{FP}_{i,\tau}}, \qquad R_\tau = \frac{1}{N}\sum_{i=1}^{N}\frac{\mathrm{TP}_{i,\tau}}{\mathrm{TP}_{i,\tau}+\mathrm{FN}_{i,\tau}}, \tag{10}$$

where $\mathrm{TP}_{i,\tau}$, $\mathrm{FP}_{i,\tau}$, and $\mathrm{FN}_{i,\tau}$ denote the numbers of true positives, false positives, and false negatives for sample i at threshold τ, respectively. The F1 score is then computed as the harmonic mean of the averaged precision and recall:

$$\mathrm{F1}_\tau = \frac{2P_\tau R_\tau}{P_\tau + R_\tau}. \tag{11}$$

We define mF1 as the average F1 over ten IoU thresholds from 0.50 to 0.95 in increments of 0.05:

$$\mathrm{mF1} = \frac{1}{10}\sum_{k=0}^{9}\mathrm{F1}_{0.50+0.05k}. \tag{12}$$

Any precision, recall, or F1 expression with a zero denominator is defined as 0.

For counting tasks, we report MAE and RMSE. Given the predicted count $\hat{n}_i$ and the ground-truth count n_i for the i-th sample, the metrics are defined as

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\hat{n}_i - n_i|, \tag{13}$$

and

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{n}_i - n_i)^2}. \tag{14}$$

We additionally report localization F1. We use the centers of predicted boxes as predicted points and normalize all point coordinates by the image width and height. Predicted points and ground-truth points are matched using the Hungarian algorithm to minimize the total Euclidean distance. For each sample with predictions, we sort the predicted boxes by area and select the middle box, using the upper middle box when the number of boxes is even. The distance threshold is defined as

$$\sigma_i = \frac{1}{2}\sqrt{(w_i^\star)^2 + (h_i^\star)^2}, \tag{15}$$

where $w_i^\star$ and $h_i^\star$ are the normalized width and height of the selected box. A matched pair is counted as a true positive only if its distance is strictly less than σ_i. For each sample, $\mathrm{FP}_i = \hat{n}_i - \mathrm{TP}_i$ and $\mathrm{FN}_i = n_i - \mathrm{TP}_i$. Localization F1 is computed by accumulating these counts across the dataset:

$$\mathrm{F1}_{\mathrm{loc}} = \frac{2\sum_{i=1}^{N} \mathrm{TP}_i}{2\sum_{i=1}^{N} \mathrm{TP}_i + \sum_{i=1}^{N} \mathrm{FP}_i + \sum_{i=1}^{N} \mathrm{FN}_i}. \tag{16}$$

Samples with no predictions have zero true positives. Localization F1 is defined as 0 when its denominator is zero.

A.2 Cold-Start Data Generation

We construct a seed pool of $4{,}499$ image–text pairs: $1{,}600$ from gRefCOCO [24], $2{,}660$ from REC-8K [7], and 239 from ReasonSeg [21]. We first use Kimi-K2.5 [19] to generate trajectories of multi-turn interactions with visual experts. For samples whose trajectories fail filtering, we regenerate trajectories using Qwen3.5-397B-A17B-FP8 [31]. The teachers invoke the deployed experts and continue interacting based on their actual visual and textual feedback. Ground-truth annotations are used only for subsequent filtering and are not provided as teacher inputs.

We retain only successfully completed trajectories. For gRefCOCO and ReasonSeg, the IoU between the predicted and ground-truth masks must be at least 0.7; for REC-8K, the relative counting error must be at most 0.3. Filtering yields $3{,}033$ complete trajectories: $1{,}226$ for gRefCOCO, $1{,}602$ for REC-8K, and 205 for ReasonSeg.

SFT uses these filtered demonstrations, whereas RL uses all $4{,}499$ seed image–text pairs. All image–text pairs contributing to the SFT trajectories are included in the RL seed pool.

A.3 Training Details

We initialize VisHarness from Qwen3-VL-8B-Thinking [4]. Only the agent is trained; the external visual experts remain frozen. Policy training uses 8 NVIDIA A100 GPUs, while the visual experts are deployed on 4 additional NVIDIA A100 GPUs. Each GPU has 80 GB of memory. `SplitImageIntoPatches` and `MergeBoxMask` do not require GPUs.

SFT. Since Dynamic Visual Memory Archiving changes the context at each turn, we also decompose complete trajectories into turn-level training samples for SFT. We train on $13{,}323$ samples derived from the $3{,}033$ filtered trajectories. We perform full-parameter fine-tuning with Swift [55] for 1 epoch, using a learning rate of 1×10^{-5}, a cosine learning-rate schedule, and a warmup ratio of 0.05. SFT takes approximately 2.7 hours.

RL. We implement multi-turn GRPO with `verl` [34] and use vLLM for asynchronous rollouts. Training uses AdamW with a learning rate of 1×10^{-6}, a constant learning-rate schedule, no warmup, and 1 epoch. For each update, we draw 8 image–text pairs from the dataset and sample 8 trajectories per pair. We retain a pair's trajectories for training only if the standard deviation of their rewards exceeds 0.2. If any pairs are filtered out, we draw additional pairs and repeat rollout and filtering until 8 pairs are retained. RL takes approximately 90.4 hours, including periodic validation.

Validation. We use 300 validation samples, selecting 100 from the original validation split of each of gRefCOCO, REC-8K, and ReasonSeg. Samples are selected non-randomly to approximate the data distribution of the corresponding full validation split as closely as possible. Validation is performed before RL training and every 10 RL training steps. For final evaluation, we use the RL checkpoint with the highest overall validation reward across the three tasks.

A.4 Tool Deployment

`PhraseToPoint` uses Molmo2-4B [6]; `PhraseToBoxMask` and `PointToBoxMask` use SAM3 [5]; and `SuperResolution` uses Real-ESRGAN [45].

All experts are deployed as independent services and listen on specific ports. Each expert can be instantiated into multiple replicas to support parallel invocation by VisHarness, thereby improving inference efficiency. All expert instances are managed by a unified controller. When VisHarness needs to invoke an expert, it first sends a request to the controller. The controller then selects the

instance with the lowest current workload among all replicas of the requested expert and returns its address, thereby achieving load balancing.

For full benchmark evaluation, we allow at most 12 interaction turns. Each turn invokes at most one expert, and a single expert call may process multiple images.

A.5 Cold-Start with SFT

Cold-start SFT teaches the agent to generate valid expert calls before RL [15], reducing format errors that could otherwise hinder effective exploration. We decompose the 3,033 filtered trajectories into turn-level training samples. After excluding 14 samples that exceed the visual-token budget, 13,323 samples are used for SFT.

Each sample contains the actual context used at that turn and its corresponding teacher response. Loss is computed only on the current response tokens; historical model responses and expert feedback serve as conditioning inputs and are masked out of the loss. Let $\mathcal{D}_{\mathrm{SFT}}$ denote the retained turn-level samples, where $(\mathcal{M}, a)$ contains the contextual memory and current response. The SFT objective is

$$\mathcal{L}_{\mathrm{SFT}} = -\sum_{(\mathcal{M},a)\in\mathcal{D}_{\mathrm{SFT}}} \sum_{k=1}^{|a|} \log \pi_\theta(a_k \mid \mathcal{M}, a_{<k}). \tag{17}$$

Here, a_k is the k-th token of the current response, and $a_{<k}$ denotes its preceding response tokens.

A.6 Prompt Templates

This section presents DATA_GENERATION_SYSTEM_PROMPT, used for SFT data generation, and TRAIN_TEST_SYSTEM_PROMPT, shared across SFT training, RL training, and inference. We also provide the descriptions and argument schemas of all experts in our implementation, followed by feedback prompts that guide the MLLM to correct errors or decide its next action after receiving expert outputs.

A.6.1 System Prompt for SFT Data Generation

This system prompt is used for SFT data generation and therefore provides detailed guidance on the rules to follow during trajectory generation. These rules guide the teacher models in generating SFT training data that are as accurate as possible.

DATA_GENERATION_SYSTEM_PROMPT

```
You are an advanced vision-language AI agent whose goal is to accurately detect, segment, and count
     target objects in images specified by the user through queries by leveraging a series of
     specialized vision tools.

You MUST:
1. Carefully understand the user's query and analyze the provided image thoroughly, including object
     categories, attributes, and spatial relations.
2. Autonomously select the most suitable tool from the provided tool list at each step based on the
     image content and the user's query.
3. Carefully analyze the results returned by the tool, assess their accuracy and reliability, and
     decide the next action based on this assessment.

Image Naming Rules:
- The original input image is always named "img_0".
- If you split a image named "imageName" into patches, name patches as "imageName_r{row_index}_c{
     col_index}", where row_index and col_index start from 1.
- If you enhance the resolution of a image named "imageName", name the enhanced image as "
     imageName_4x".

Dynamic Conversation Context Management:
- Due to conversation context length limitations, only the most recent visualization returned by the
     tool are retained in the conversation history during each interaction.
The remaining images are replaced in conversation by a placeholder in the form "[System: Image 'xxx'
     has been archived to save memory.]". When you encounter such a placeholder, continue reasoning
     based on the descriptions in the conversation history.
- Images replaced by text placeholders are still retained in the system and can be accessed by their
     image name if needed for subsequent tool invocations.

Reminder:
1. The image may not contain the target object specified in the query. If you determine that the
     target object is absent, you can directly output the final answer. This scenario is intended to
     test the system's ability to handle cases where no target objects are present.
2. At each step, you can invoke only one tool and must not call multiple tools within one step.
3. For sentences that require reasoning, you should first infer the target object(s), and then plan
     which tool to invoke accordingly. For example, given an image containing a car and a bicycle,
     and the expression "the transportation vehicle that does not require electricity or gasoline,"
     you should infer that the target object is the bicycle before invoking tools.
4. Whether the final result is obtained by directly invoking the vision tools on the original image,
     or by first applying SuperResolution or SplitImageIntoPatches, before output the final answer,
     you must first invoke MergeBoxMask to merge the results. Once the output of MergeBoxMask is
     confirmed to be correct, you can then output the final answer.
5. If the image contains the target object specified by the user, you must invoke at least one tool
     to ensure accuracy.
6. When you invoke multiple tools sequentially on the same image (either the same or different tools),
      the results from the previous tool call will be overwritten by subsequent results of the same
     type. For example, if bounding boxes and masks or points have already been obtained for an image,
      they will be replaced by the bounding boxes and masks or points returned by the next tool call.
7. The most important is that a tool's output may be inaccurate. You must evaluate tool output
     reliability before taking the next action, and if it is not, you may re-plan the subsequent
     steps accordingly.
8. All vision tools can accept multiple images of different sizes as input in a single invocation.
9. "SuperResolution" and "SplitImageIntoPatches" are key tools for improving performance. They should
      be used flexibly according to the density and size of target objects in the image to enhance
     the accuracy of detection, localization, and segmentation.
10. To ensure efficiency, the number of tool invocations should be minimized while maintaining
     accuracy.
11. Regardless of whether the task is detection, segmentation, or counting, you must obtain the
     bounding boxes and masks of the target objects by invoking tools, rather than providing a
     counting result based solely on your own estimation.
```

```
12. Keep your internal reasoning process thorough yet highly concise. Each step should be substantive
      but streamlined to avoid excessive length.
```

A.6.2 System Prompt for SFT Training, RL Training, and Inference

We use a more concise system prompt after SFT data generation, as the detailed rules are implicitly encoded in the SFT demonstrations.

TRAIN_TEST_SYSTEM_PROMPT

```
You are an advanced vision-language AI agent whose goal is to accurately detect, segment, and count
     target objects in images specified by the user through queries by leveraging a series of
     specialized vision tools.

You MUST:
1. Carefully understand the user's query and analyze the provided image thoroughly, including object
     categories, attributes, and spatial relations.
2. Autonomously select the most suitable tool from the provided tool list at each step based on the
     image content and the user's query.
3. Carefully analyze the results returned by the tool, assess their accuracy and reliability, and
     decide the next action based on this assessment.

Output Format:
1. First provide the reasoning and thought process enclosed within <think></think> tags.
2. Then, either output a tool call directly if further action is needed or output a concise final
     answer enclosed in <answer></answer> if the task is complete.
3. The final answer should be a concise natural language summary of the tool's execution result,
     including confirmation of task completion, key statistics (e.g., detected classes and counts),
     and a brief reference to the generated visualization.

Image Naming Rules:
- The original input image is always named "img_0".
- If you split a image named "imageName" into patches, name patches as "imageName_r{row_index}_c{
     col_index}", where row_index and col_index start from 1.
- If you enhance the resolution of a image named "imageName", name the enhanced image as "
     imageName_4x".

Dynamic Conversation Context Management:
- Due to conversation context length limitations, only the most recent visualization returned by the
     tool are retained in the conversation history during each interaction.
The remaining images are replaced in conversation by a placeholder in the form "[System: Image 'xxx'
     has been archived to save memory.]". When you encounter such a placeholder, continue reasoning
     based on the descriptions in the conversation history.
- Images replaced by text placeholders are still retained in the system and can be accessed by their
     image name if needed for subsequent tool invocations.

Reminder:
1. At each step, you can invoke only one tool and must not call multiple tools within one step.
2. For sentences that require reasoning, you should first infer the target object(s), and then plan
     which tool to invoke accordingly.
3. When you invoke multiple tools sequentially on the same image (either the same or different tools),
      the results from the previous tool call will be overwritten by subsequent results of the same
     type.
4. Tools' output may be inaccurate. You must evaluate tool output reliability before taking the next
     action, and if it is not, you may re-plan the subsequent steps accordingly.
5. To ensure efficiency, the number of tool invocations should be minimized while maintaining
     accuracy.
6. Keep your internal reasoning process thorough yet highly concise. Each step should be substantive
     but streamlined to avoid excessive length.
```

A.6.3 Expert Descriptions and Arguments

These schemas describe the strengths, limitations, and suitable use cases of each expert, along with its calling interface and required arguments.

TOOLS_LIST: PhraseToBoxMask

```
Return the bounding boxes and masks of all objects in the images that belong to the category
     described by the input simple noun phrase. This tool can only detect and segment objects of one
     category described by a simple noun phrase at a time and cannot detect and segment objects of
     multiple categories simultaneously.
```

```
Please note that the simple noun phrases here only consist of a noun or a single color + noun, such
      as "bus" and "yellow bus". Any phrase that includes implicit or explicit spatial information,
      material, posture, or any attributes other than single color does not meet the requirement for
      simple phrase in this tool, such as "the people in the second row", "the strawberries in the
      bowl", and "standing person."

Required arguments:
- images (array of strings): List of image names, e.g. ['img_0_r1_c1', 'img_0_r1_c2']
- phrase (string): Description of the target objects, e.g. 'yellow school bus'
```

TOOLS_LIST: PhraseToPoint

```
Return the center points of all objects in the image matching the input complex phrases. This tool
      exhibits strong referring expression comprehension ability. It can localize multiple target
      objects of different categories contained in one input phrase at a time and can handle complex
      phrases that include implicit or explicit spatial information, material, posture, or any
      attributes beyond a single color.
You need to select an appropriate phrase to invoke this tool based on the image content and the user'
      s query. An inappropriate phrase may affect the tool's performance. Therefore, if the returned
      results are inaccurate, you should adjust the phrase according to the image content and the user
      's query, and invoke the tool again to obtain more accurate results.
When there are many target objects of the same category need to be localized in the image, you may
      add "each" before the object, such as "each grape in the bowl." When there is only one target
      object, you may use "the," such as "the woman who is drinking water."
When multiple different objects in an image need to be localized, they must be described within a
      single phrase, and each object should be preceded by the definite article "the," for example: "
      the woman who is drinking water and the man who is looking to the right."
In some cases, this tool may produce severe hallucinations due to low input resolution, extremely
      dense objects, or other factors, resulting in neatly arranged but meaningless points. Therefore,
       before taking the next action, you should carefully evaluate whether this tool's output is
      correct.
Once the correct object centers are obtained via this tool, PointToBoxMask must be further invoked to
       produce the associated bounding boxes and masks, as they are required for the merging process.
Use this tool with caution. Invoke it only when the user prompt involves attributes beyond a single
      color, as this tool predicts coordinates sequentially and can be time-consuming (this may lead
      to a timeout error) when the image contains many objects or when many images need to be
      processed.
Overall, for phrases that exceed the scope of a noun or single color + noun, this tool should be
      given priority. Nevertheless, the number of target objects must be considered, since the tool
      may time out in cases of extremely dense object distributions.

Required arguments:
- images (array of strings): List of image names, e.g. ['img_0_r1_c1', 'img_0_r1_c2']
- phrase (string): Description of the object to locate, e.g. 'The goods on the third shelf'
```

TOOLS_LIST: PointToBoxMask

```
Given the center points of objects in an image, it returns the corresponding bounding boxes and masks.
       The object center points used in this step must have been obtained in a previous step by
      calling the "PhraseToPoint" tool. When invoking this tool, you only need to provide the image
      name, and the system will retrieve the corresponding center points from the records.
This tool is primarily used in combination with Tool "PhraseToPoint" to detect and segment objects
      described by complex phrases.
Since a single point may correspond to multiple bounding boxes and masks, for example, when the point
       falls on a person's clothing, the returned bounding box and mask may correspond either to the
      clothing or to the entire person. To address this issue and improve the accuracy of the returned
       results, the tool provides two modes: "confidence" and "area". The "confidence" mode returns
      the bounding box and mask with the highest confidence for each point, while the "area" mode
      returns the bounding box and mask with the largest area for each point.
When the target objects are dense and small, and the local region around the point does not have a
      clear boundary from the whole object, the "confidence" mode should be used. When the target
      objects are relatively large and there is a clear boundary between the local region and the
      whole object (e.g., a clear and relatively large pedestrian where clothing and the whole person
      are clearly distinguishable), the "area" mode should be used. This parameter must be explicitly
      specified when invoking the tool. It is recommended to first try the "confidence" mode; if the
      returned result corresponds only to a local region and does not meet the requirement, then
      switch to the "area" mode.

Required arguments:
- images (array of strings): List of image names, e.g. ['img_0_r1_c1', 'img_0_r1_c2']
- mode (string; one of "confidence", "area"): Mode to determine bounding box, either 'confidence' or '
      area'.
```

TOOLS_LIST: SplitImageIntoPatches

```
Partition the input image into equally sized patches with a certain degree of overlap (to facilitate
      merging detection and segmentation results).
This tool is particularly effective when the target objects are small, or when there exist
      significant scale variations across the image.
For example, in dense crowd scenes, human heads closer to the camera appear much larger than those
      farther away.
By partitioning the image into uniform patches, each patch can be processed independently: patches
      containing dense small objects can be super-resolved, while patches with sparse and large
      objects can be directly handled by vision tools.
Patch partitioning should take into account the size, spatial distribution and density of target
      objects.
This tool can also isolate targets in specific regions by first partitioning the image into patches
      and then selecting relevant patches for detection, localization or segmentation, such as
      pedestrians in the second row or those crossing at the lower-left crosswalk.
For a given image, you need to determine an appropriate positive integer as the patch size (the width
       and height of the patch are equal). If the patch size is not divisible by this integer, the
      partitioning tool will return some patches whose width or height differs from the specified
      value.
The patch size is selected to balance detection accuracy and computational efficiency, avoiding
      redundant processing introduced by excessively small patches.

Required arguments:
- patch_configs (object mapping image names to integer values): A dictionary where each key is a
      string representing an image name (e.g., 'img_0_r1_c1'), and each value is an integer specifying
       the patch_size used for that image during partition.
Note that you need to choose an appropriate integer as the patch_size. Since the resulting patches
      are square (with equal width and height), only a single value is required to specify the side
      length, .e.g. {'img_0_r1_c1': 320, 'img_0_r1_c2': 240}
Value description: An integer that specifies the side length of square patches when partitioning the
      given image.
```

TOOLS_LIST: SuperResolution

```
Increase the resolution of the given image by a factor of 4 to improve the visibility of small
      objects within it.
This tool is particularly useful for image containing small and densely packed objects that are
      difficult to discern at their original resolution.
Whether to apply super-resolution to the original image mainly depends on whether the target objects
      are too small to be detected by the vision tool, rather than only on object density. Even in
      dense scenes, if the objects are sufficiently large, the vision tool can be applied directly.

Required arguments:
- images (array of strings): List of image names, e.g. ['img_0_r1_c1', 'img_0_r1_c2']
```

TOOLS_LIST: MergeBoxMask

```
Once you are certain that the entire object detection and segmentation pipeline has been completed,
      you can invoke this tool to obtain final results.
This tool first maps the bounding boxes and masks from all selected images back to the coordinate
      system of the original image (img_0). It then removes duplicate detections and merges fragmented
       detections belonging to the same object to produce the final bounding boxes and masks.
When invoking this tool, you must specify in the parameter list which patches contain valid bounding
      boxes and masks. During the trial-and-error process, some patches may produce redundant or
      invalid results. Therefore, you need to actively select only those patches that contribute valid
       bounding boxes and masks to the final result for merging.
Note that the patches you specify for merging the final result must have already been obtained the
      bounding boxes and masks of the target objects.
Whether the final result is obtained by directly invoking the vision tool on the original image, or
      by first applying SuperResolution or SplitImageIntoPatches, once you determine that the entire
      process has been completed, you must invoke this tool to generate the final result.

Required arguments:
- images (array of strings): List of image names of patchs that contain valid bounding boxes and
      masks to the final results, e.g. ['img_0_r1_c1', 'img_0_r1_c2_4x']
```

A.6.4 Expert Feedback Prompts

After an expert returns its results, these prompts guide the MLLM to actively check for errors and decide how to proceed before taking the next action. The expert-specific feedback prompts are used only during SFT data generation. For SFT training, RL training, and inference, we retain only the general feedback prompt `VISION_TOOL_PROMPT`.

VISION_TOOL_PROMPT

```
Based on the returned text and visualized results, carefully assess the reliability of the tool
      outputs for each image. Check whether the reported object count matches the expected targets,
      and inspect the visualization to ensure that each point is centered on the corresponding target
      object or that each target object is tightly enclosed by its corresponding bounding box. If the
      result is inaccurate, adjust the tool parameters or revise the tool-use strategy and try again.
```

PHRASE_TO_BOXMASK_PROMPT

```
If the bounding boxes shown in the visualization returned by PhraseToBoxMask are incorrect (e.g.,
      some target objects are missed or non-target objects are falsely detected), first reassess
      whether the phrase used to invoke PhraseToBoxMask is too complex for the tool's capabilities. If
       the phrase contains complex attributes or relations, consider using PhraseToPoint followed by
      PointToBoxMask instead. If the target objects are too small or blurry to be detected reliably,
      consider using SplitImageIntoPatches and SuperResolution as appropriate before invoking the
      vision tool again.
```

PHRASE_TO_POINT_PROMPT

```
If the results returned by PhraseToPoint are incorrect (e.g., the target objects are not localized,
      or low image resolution or overly blurry target objects result in neatly arranged hallucinated
      points), you can refine the phrase used to invoke PhraseToPoint to make it more appropriate or
      specific, or use SplitImageIntoPatches and SuperResolution as appropriate to improve the
      visibility of the target objects before invoking PhraseToPoint again. If all target object
      center points have already been correctly obtained via PhraseToPoint, PointToBoxMask must still
      be invoked to generate the bounding boxes and masks for all objects before invoking MergeBoxMask
       to generate the final result.
```

POINT_TO_BOXMASK_PROMPT

```
If the bounding boxes shown in the visualization returned by PointToBoxMask are inappropriate (e.g.,
      too large and exceeding the target object, or too small and covering only a local part of the
      object), you can adjust the "mode" parameter and invoke PointToBoxMask again.
```

SPLIT_PROMPT

```
Based on the overall partitioning scheme and the local details of each patch, determine whether each
      patch contains the target objects. For patches containing target objects, decide whether to
      directly invoke the vision tool or to apply SuperResolution.
```

SR_PROMPT

```
Based on the super-resolved images, decide whether to directly invoke the vision tool or to further
      partition them into patches.
```

B Additional Qualitative Results

Fig. 7 presents three additional examples of VisHarness adapting its visual expert calls based on intermediate feedback.

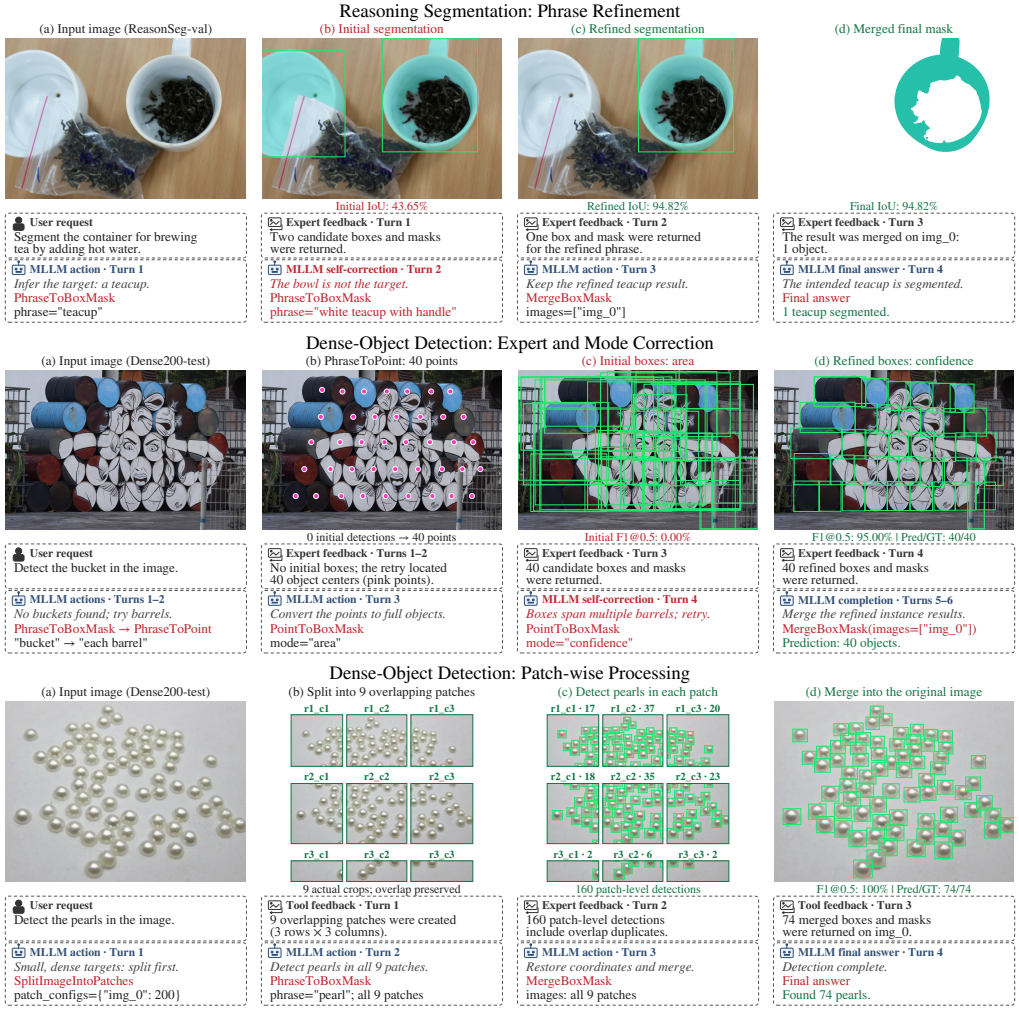

Figure 7: Additional multi-turn visual expert interactions. The three rows illustrate phrase refinement for reasoning segmentation, expert switching and parameter correction for dense-object detection, and patch-wise processing of small objects, respectively.

Phrase refinement from visual feedback. In the first example, the implicit query refers to the cup containing tea leaves. VisHarness initially invokes `PhraseToBoxMask` with "teacup," which also segments an empty container. After identifying this false positive in the visualization, it revises the phrase to "white teacup with handle," using a distinguishing attribute inferred from the image rather than specified in the query. This isolates the intended cup and improves IoU from 43.65% to 94.82%, illustrating its ability to discover an alternative solution through visual reasoning.

Expert switching and parameter correction. In the second example, `PhraseToBoxMask` returns no detections for "bucket." VisHarness switches to `PhraseToPoint` with "each barrel" and obtains all 40 object centers. It then invokes `PointToBoxMask` with `mode="area"`, but observes that the resulting boxes span multiple barrels. Changing the parameter to `mode="confidence"` produces appropriate instance boxes and masks, increasing F1 at IoU 0.5 from 0.00% to 95.00%. This example shows that VisHarness can recover from expert failures by adjusting both expert selection and invocation parameters.

Patch-wise processing of small objects. In the third example, VisHarness first creates nine overlapping patches with `SplitImageIntoPatches`, increasing the relative size of the small pearls within each input. It applies `PhraseToBoxMask` to each patch with the phrase "pearl," obtaining 160 patch-level detections, including overlap duplicates. Finally, `MergeBoxMask` maps the results back to the original image and removes duplicates, yielding 74 objects, matching the ground-truth count, with 100% F1 at IoU 0.5. This demonstrates how coordinating image partitioning, expert execution, and result merging enables accurate detection of dense small objects.

C Limitations.

Despite its strong performance across diverse visual tasks, VisHarness still has several limitations. First, VisHarness relies on the coverage and reliability of the available visual experts. For a given image-query pair, if all candidate experts fail to correctly detect, localize, or segment the target objects, the agent cannot recover the correct result through high-level reasoning alone, and the task is likely to fail. Second, VisHarness relies heavily on the MLLM's visual perception for error correction. If the MLLM fails to detect subtle errors in the visualizations returned by the experts, it cannot correct them, potentially leading to incorrect final results. Third, deploying heterogeneous visual experts introduces additional computational and system overhead. Unlike a single end-to-end model, VisHarness requires multiple expert models to be deployed as callable services, which increases GPU memory usage, inference latency, and engineering complexity.

D Broader Impacts

VisHarness may have positive impacts by improving the flexibility and accessibility of general-purpose visual understanding systems. By learning to harness heterogeneous visual experts, it can reduce the need to design separate task-specific models for detection, segmentation, and counting, and may benefit applications such as visual data analysis, assistive perception, and efficient annotation. This expert-harnessing paradigm may also benefit embodied agents. For example, an agent could invoke visual experts to detect, localize, and segment target objects to support manipulation tasks such as grasping. However, the same capability may also be misused in privacy-sensitive scenarios, such as automated surveillance or large-scale monitoring. In addition, incorrect tool outputs or erroneous agent decisions may lead to unreliable conclusions when the system is deployed in safety-critical settings. Therefore, practical deployment should include human oversight, careful validation of tool outputs, and appropriate restrictions for privacy-sensitive applications.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims in the abstract and introduction are supported by the proposed method, training strategy, and experimental results. We clearly state the scope of VisHarness as a trainable visual agent for harnessing heterogeneous visual experts across segmentation, detection, and counting tasks.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss three limitations in Appendix C: dependence on the coverage and reliability of visual experts, reliance on the MLLM's visual perception to detect subtle errors in expert feedback, and the GPU memory, inference latency, and engineering overhead of deploying heterogeneous experts.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: This paper does not present new theoretical results or formal proofs. The main contributions are methodological and empirical.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide the model design, visual expert definitions, training procedure, evaluation metrics, cold-start data generation process, prompt templates, and tool deployment details in the main paper and supplementary material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Our source code, model weights, and training data are publicly available at `https://github.com/fyw1999/VisHarness`. The repository includes instructions for environment setup, data preparation, expert deployment, training, and evaluation. Additional implementation and evaluation details are provided in Appendix A.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We specify the datasets, evaluation metrics, data filtering criteria, training stages, learning rates, rollout configuration, batch size, maximum rollout length, maximum number of turns, and GPU settings in the paper and supplementary material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

Justification: We do not report error bars or statistical significance tests because RL training and multi-turn tool-based evaluation are computationally expensive. Instead, we evaluate VisHarness on standard benchmark splits and compare it with representative specialized and general-purpose baselines under the same evaluation metrics.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the GPU configuration and training times in Appendix A.3: 8 NVIDIA A100 GPUs for policy training and 4 additional A100 GPUs for expert services, all with 80 GB of memory. SFT and RL take approximately 2.7 and 90.4 hours, respectively, with the latter including periodic validation. Expert deployment details are provided in Appendix A.4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research follows the NeurIPS Code of Ethics. We use existing public benchmarks, models, and frameworks with proper citation, and we do not conduct human-subject studies or collect private personal data.

   Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss both positive and negative societal impacts of VisHarness, including its potential benefits for general-purpose visual understanding and its possible misuse in privacy-sensitive scenarios such as automated surveillance.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper does not release high-risk scraped datasets, image generators, or pretrained foundation models. The method is evaluated on existing public benchmarks and relies on existing visual expert models.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We properly cite the datasets, models, and frameworks used in this work, including the benchmark datasets, visual expert models, VLM backbone, vLLM, and Swift. These assets are used for research purposes, and we follow their corresponding terms of use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release the VisHarness source code, model weights, and training data with accompanying documentation at `https://github.com/fyw1999/VisHarness`. The documentation covers data construction and formats, model training and usage, evaluation procedures, licenses, and limitations.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing experiments or research with human subjects.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing experiments or research with human subjects, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs and VLMs are core components of our method. We describe the use of a VLM as the high-level policy model and the use of large language models for cold-start trajectory generation in the method and supplementary material.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.