ZoomEye: Enhancing Multimodal LLMs with Human-Like Zooming Capabilities through Tree-Based Image Exploration

Anonymous ACL submission

Abstract

Multimodal Large Language Models (MLLMs) have demonstrated impressive capabilities in vision-language understanding. Recently, with the integration of test-time scaling techniques, these models have also shown strong potential in visual reasoning. However, most existing reasoning approaches remain text-level in nature: MLLMs are prompted to explore various combinations of textual tokens via their underlying language model, while the visual input remains fixed throughout the reasoning process. This paradigm limits the model's ability to fully exploit rich visual information, particularly when dealing with images containing numerous fine-grained elements. In such cases, vision-level reasoning becomes crucial—where models dynamically zoom into specific regions of the image to gather detailed visual cues necessary for accurate decision-making. In this paper, we propose Zoom Eye, a training-free, model-agnostic tree search algorithm tailored for vision-level reasoning. Zoom Eye treats an image as a hierarchical tree structure, where each child node represents a zoomed-in subregion of its parent, and the root corresponds to the full image. The algorithm enables MLLMs to simulate human-like zooming behavior by navigating from root to leaf nodes in search of task-relevant visual evidence. We experiment on a series of elaborate high-resolution benchmarks and the results demonstrate that Zoom Eye not only consistently improves the performance of a series of MLLMs with large margin (e.g., InternVL2.5-8B increases by 15.71% and 17.69% on HR-Bench) but also enables small 3-8B MLLMs to outperform strong large models such as GPT-4o.

1 Introduction

By integrating powerful language models (Touvron et al., 2023; Yang et al., 2024) with visual encoders (Radford et al., 2021; Sun et al., 2023; Zhai et al., 2023), Multimodal large language models (MLLMs) are able to jointly process textual and visual inputs, achieving impressive performance in vision-language understanding (Zhao et al., 2024a; Bai et al., 2023; Chen et al., 2024b; Li et al., 2024). Recently, drawing on test-time scaling techniques that enhance reasoning abilities in LLMs, such as OpenAI-o1 (Jaech et al., 2024) and DeepSeek-R1 (Guo et al., 2025), a series of literature tries to investigate these reasoning techniques in MLLMs to further improve the visual reasoning capabilities (Xu et al., 2024; Dong et al., 2024; Yao et al.,

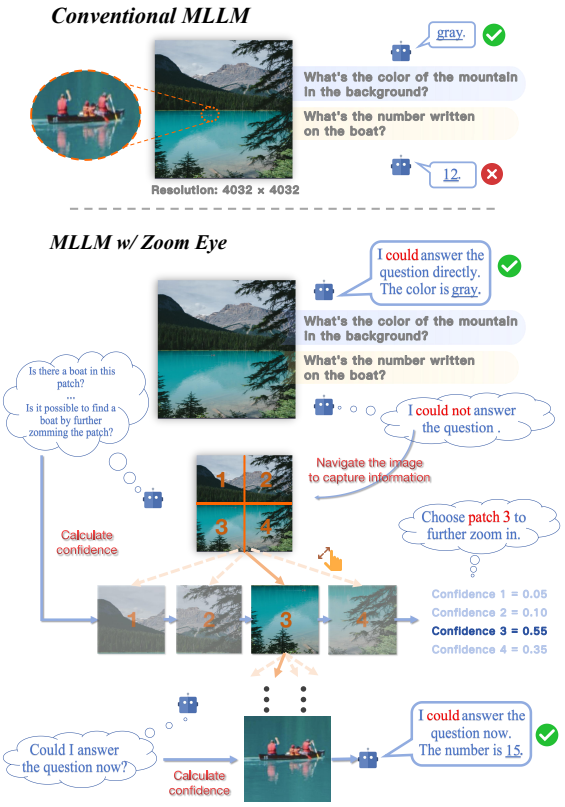

Figure 1: **Top**: When dealing with a high-resolution image, MLLMs effectively perceive the dominant objects but often fail to recognize finer details, highlighting the need for vision-level reasoning. **Bottom**: Applied with Zoom Eye, MLLMs could perform vision-level reasoning, allowed to explore the image details until they can answer the question.

2024a; Shen et al., 2025; Meng et al., 2025)

However, these methods predominantly operate at the textual level, leveraging the generative capacity of the underlying language model without modifying the perception of the image itself. That is, the visual input remains static throughout the reasoning process, restricting the model's ability to process fine-grained visual content, especially on an elements-rich high-resolution image. As illustrated in the top of Figure 1, for the same image, the MLLM accurately recognizes the dominant object whereas it struggles to perceive the detailed one. This gap highlights the need for vision-level reasoning, where the model actively interacts with the image by zooming in and out to selectively attend to informative regions, as demonstrated in the bottom of Figure 1, much like how humans visually process complex scenes. A similar vision-level zooming mechanism has been adopted in the closed-source OpenAI-o3 (OpenAI, 2025). In contrast, our goal is to develop an open-source vision-level reasoning method, making this capability accessible to the broader research community.

When viewing a high-resolution image, humans typically start with a global scan, then gradually zoom into areas of interest for closer inspection (Figure 2(b)). If the desired information is not found, they zoom out and explore alternative regions (as shown in Figure 2 (c)). Inspired by this, structuring an image as a tree is highly logical for simulating similar actions in an MLLM: the root denotes the full image, each child node corresponds to a zoomed-in sub-region of its parent, and deeper nodes indicate higher zoom levels. This hierarchical representation, combined with a search algorithm, allows models to (1) explore fine-grained regions (*node lookahead*) and (2) return to the previous view to inspect other regions (*node backtracking*). Similar tree-based search strategies have shown strong performance in text-based LLM reasoning(Yao et al., 2024b; Hao et al., 2023; Feng et al., 2023; Zhu et al., 2023).

In this paper, we propose Zoom Eye, a tree search algorithm for vision-level reasoning, which navigates MLLMs in the dense image context by the hierarchical and visual nature of images (**contribution #1**). This method simulates the actions of zooming in and out to inspect image details and seek out crucial information. Given a question, the adopted MLLM first identifies the pertinent objects. We then introduce two types of *confidence values* by prompting the MLLM to recognize the presence of these relevant objects. These *confidence values* are used to prioritize each candidate node during the tree search, determining the sequence of node selection. The search concludes based on a stopping criterion when the MLLM can confidently answer the question. This process is illustrated in the bottom part of Figure 1. Finally, the MLLM formulates a final response based on the visual information gathered during the search.

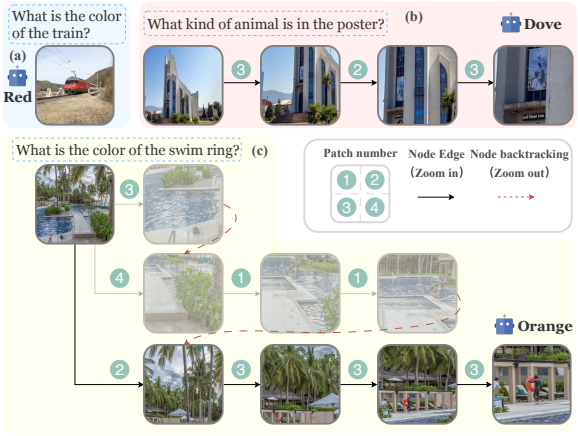

Figure 2: Zoom Eye enables MLLMs to **(a)** answer the question directly when the visual information is adequate, **(b)** zoom in gradually for a closer examination, and **(c)** zoom out to the previous view and explore other regions if the desired information is not initially found.

We adapt Zoom Eye to a series of mainstream MLLMs, including Qwen2.5VL (Bai et al., 2025), LLaVA-v1.5 (Liu et al., 2024a), LLaVA-OneVision (Li et al., 2024), InternVL2.5 (Chen et al., 2024a), and evaluate them on a suite of elaborate high-resolution visual understanding benchmarks. Equipped with Zoom Eye, all evaluated models achieve substantial performance improvements compared to the baseline (**contribution #2**).

Additionally, our analysis also reveals certain deficiencies in visual understanding exhibited by these models, which we detail in §4.3 (**contribution #3**). Addressing these limitations is part of our future work. More importantly, as discussed in §4.4.1, we observe a vision-level test-time scaling phenomenon analogous to what has been observed in text-based LLMs: performance consistently improves with an increasing number of search steps. This finding suggests that vision-level reasoning benefits from deeper exploratory search and opens new avenues for scaling MLLM inference beyond static image perception (**contribution #4**).

2 Preliminary

In this section, we describe briefly the prevalently adopted image preprocessing methods and image-text input ways of MLLMs.

Image preprocessing. For a given image $\mathbf{I}$, a **naive** processing style is to simply resize it to a preset fixed resolution and then feed it into an vision encoder to generate visual representations. This could be formulated as: $\mathbf{v} = \mathcal{F}(R(\mathbf{I})) = (v_1, v_2, \ldots, v_{L_v})$, where $\mathcal{F}$ is the vision encoder, R is the resize operation, and L_v is the number of visual representations. Due to the constraints of the naive version's fixed and limited resolution, another method, known as **AnyRes**, was introduced. It divides the original image into several equal-area blocks and imposes a maximum limit, M, on the number of divided blocks. The vision encoder then independently encodes each block and the overall image. Finally, all the encoded visual representations are integrated together. This allows flexible processing of various resolutions. Denoting $\mathbf{I}^{(0)}$ as the whole image and $\{\mathbf{I}^{(1)}, \ldots, \mathbf{I}^{(a)}\}$ $(a \leq M)$ as the blocks, the AnyRes could be formulate as: $\mathbf{v} = \mathcal{F}(A(\mathbf{I})) = [\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_a]$, where A denotes the AnyRes operation and $\mathbf{v}_i = \mathcal{F}(R(\mathbf{I}^{(i)})) = (v_{(i,1)}, v_{(i,2)}, \ldots, v_{(i,L_v)})$, $i = 0, 1, \ldots, a$. It is noteworthy that the naive method can be considered a special case of AnyRes when $a = 0$.

Imga-Text joint input for MLLM. Common MLLMs link a vision encoder to the pre-trained LLM via projection or alignment modules, allowing language generation through the autoregressive capabilities of their LLM base. Specifically, given an image $\mathbf{I}$ and an input prompt $\mathbf{x}$, $\mathbf{I}$ is first encoded into a set of visual representations as described in the previous sub-section. Subsequently, these visual representations, along with the text input, are fed into the LLM base of the MLLM. Assuming the length of the output sequence and text input are L_y and L_x respectively, the probability for a MLLM Φ_θ to generate an output $\mathbf{y} = (y_1, y_2, \ldots, y_{L_y})$ conditioned on the visual input $\mathcal{F}(\cdot(\mathbf{I})) = (v_{(0,1)}, \ldots, v_{(a,L_v)})$ and the text input $\mathbf{x} = (x_1, x_2, \ldots, x_{L_x})$ is: $\Phi_\theta(\mathbf{y}|\mathcal{F}(\cdot(\mathbf{I})), \mathbf{x}) = \prod_{i=1}^{L_y} \Phi_\theta(y_i|v_{(0,1):(a,L_v)}, x_{1:L_x}, y_{1:i-1})$, where $\mathcal{F}(\cdot)$ could represent $\mathcal{F}(R)$ as naive resize or $\mathcal{F}(A)$ as AnyRes.

3 Methodology

In this section, we introduce the Zoom Eye algorithm. Firstly, we brief the general tree search algorithm. Subsequently, we elaborate on our implementation by initializing the components of the tree search algorithms in detail.

3.1 Abstraction of Tree Search

Tree node. Typically, a node in the tree structure comprises the following attributes:(1) **id**: The unique identifier of the node. (2) **depth**: Represents the level of the node within the tree. (3) **value**: Used to store numeric or textual data in the node. (4) **children**: A list of references to the node's children nodes, which facilitates traversal of the tree structure. (5) **Other custom attributes**

Tree search. The abstraction of the tree search algorithm could be modeled as a tuple $(T, Q, \mathcal{R}, \mathcal{S})$, where T is the tree structure consisting of a set of nodes, Q is a container that holds all the nodes that might be accessed in the next search step, $\mathcal{R}$ is a ranking function used to select the highest priority node based on the used search algorithm, and $\mathcal{S}$ represents the stopping criterion. The abstract search process is shown in Algorithm 1.

Alg. 1 Abstraction of Tree Search Algorithm

Require: $T, Q, \mathcal{R}, \mathcal{S}$
Initialize Q as the empty queue {}
Q.append(T.root)
while Q is not empty **do**
  $n_t \leftarrow Q$.pop()
  if $\mathcal{S}(n_t) ==$ True **then**
    break
  $s \leftarrow n_t$.children.size
  for $j = 1, \ldots, s$ **do**
    Q.append(n_t.children[j])
  Q.sort($\mathcal{R}$)

Consider the example of a DFS search for a node with a value of 5 in the tree, in this case, $\mathcal{R}$ is a function that sorts the nodes in Q in descending order of depth, and in ascending order of id when depths are equal. Meanwhile, $\mathcal{S}$ is a function checking if a node's value equals 5.

A specific implementation of Zoom Eye search involves three key questions: 1. How to formulate the image as a tree T (§3.2). 2. How to set the ranking function $\mathcal{R}$ (§3.3). 3. How to determine the stopping criterion $\mathcal{S}$ (§3.4). Finally, we provide a description of the overall algorithm in §3.5.

3.2 Tree Representation for Image

We model the overall image as a tree T. A specific *node*, denoted as n_t, represents an image patch view $\{\mathbf{I}, \mathbf{b}_t\}$, where $\mathbf{I}$ is the image and $\mathbf{b}_t = (x_{1,t}, y_{1,t}, x_{2,t}, y_{2,t})$ is the normalized bounding box coordinates. If the size of n_t's image patch

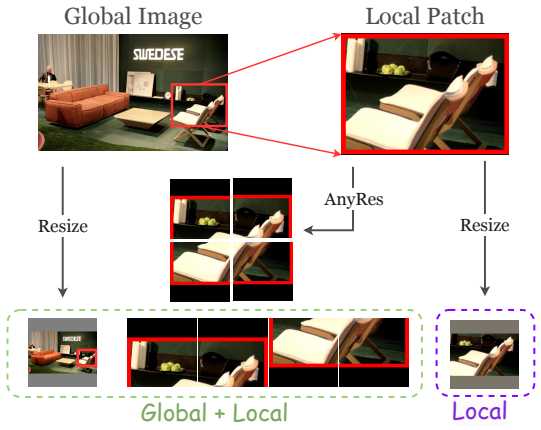

Figure 3: Two image input methods for MLLMs with distinct image processing.

exceeds the predefined resolution by the image encoder, it can be further divided into four equal-sized sub-patches, serving as its children with size 4. Nodes are recursively divided until they meet the resolution limit. At the start of the search, the root node $T.\texttt{root} = \{\mathbf{I}, (0, 0, 1, 1)\}$ representing the overall image is visited.

However, due to the detailed nature of high-resolution images and information loss from downsampling to the vision encoder's fixed resolution, MLLMs frequently struggle to accurately capture key parts of an image initially. Consequently, MLLMs should be be allowed to continuously scan and zoom into the current view (i.e., explore deeper nodes) for more focused information. In our implementation, we consider two image input methods to enable MLLMs to perceive the local patch represented by n_t: (1) Local Input: only the local patch is provided, suitable for earlier single-image input MLLMs with naive image preprocessing method (Li et al., 2023; Liu et al., 2024c,a). (2) Global+Local Input: both the global image and local patch are input, ideal for advanced MLLMs using AnyRes preprocessing method (Liu et al., 2024b; Li et al., 2024; Chen et al., 2024b). In this case, we use the visual prompt with a red rectangle to emphasize the local focus, applying naive processing to the global image and AnyRes to the local patch, as shown in Figure 3. Denoting $\mathcal{V}(n_t)$ as the final image input, we have:

$$\mathcal{V}(n_t) = \begin{cases} [\mathcal{F}(R(\mathbf{I}.\text{crop}(\mathbf{b}_t))] & \text{Local} \\ [\mathcal{F}(R(\mathbf{I})), \mathcal{F}(A(\mathbf{I}.\text{crop}(\mathbf{b}_t))] & \text{Global+Local} \end{cases} \tag{1}$$

3.3 Ranking Function

As shown in Algorithm 1, $\mathcal{R}$ is used to rank the nodes with the priority value to determine which

Alg. 2 Ranking Function & Stopping Criterion

Require: $\Phi_\theta, \mathcal{W}, \{\mathrm{p}_e, \mathrm{p}_l, \mathrm{p}_a\}, \tau, o, q_s$

function $\mathcal{R}(n_1, n_2)$ ▷ Ranking Function
  return GET PRIORITY(n_1) > GET PRIORITY(n_2)

function $\mathcal{S}(n_t)$ ▷ Stopping Criterion
  $c_a \leftarrow$ LOGITS RATIO$(n_t, \mathrm{p}_a(q_s))$
  return $c_a \geq \tau$

function GET PRIORITY(n_t)
  if $n_t.\texttt{priority}$ is None **then**
    $c_e \leftarrow$ LOGITS RATIO$(n_t, \mathrm{p}_e(o))$
    $c_l \leftarrow$ LOGITS RATIO$(n_t, \mathrm{p}_l(o))$
    $\alpha \leftarrow \mathcal{W}(n_t.\texttt{depth})$ ▷ weighted factor
    $n_t.\texttt{priority} \leftarrow \alpha \cdot c_l + (1 - \alpha) \cdot c_e$
  return $n_t.\texttt{priority}$

function LOGITS RATIO$(n_t, \mathbf{x})$
  $z_1 \leftarrow \Phi_\theta(y = \texttt{Yes} \mid \mathcal{V}(n_t),\ \mathbf{x})$
  $z_2 \leftarrow \Phi_\theta(y = \texttt{No} \mid \mathcal{V}(n_t),\ \mathbf{x})$
  $z \leftarrow (softmax(z_1, z_2)[0] - 0.5) \times 2)$
  return z ▷ $z \in (-1, 1)$

one to visit in the next step. A well-defined $\mathcal{R}$ strategically steers the search process. In Zoom Eye, we adopt the MLLM to calculate the priority value and use $\mathcal{R}$ to sort nodes by the value. Specifically, let o denote the visual cue that is crucial for answering the question, a MLLM should have the following capabilities: (1) It could perceive whether o exists within the visible view; (2) If o occupies a small area and is not clearly visible, it can leverage the common sense knowledge to infer whether o might be discerned through further zooming. Thus, we query the MLLM with two prompts $\mathrm{p}_e(o)$ and $\mathrm{p}_l(o)$ (e.g., "`Is there a` o `in the sub-patch?`", "`Is it possible to find a` o `by further zooming the sub-patch?`") to trigger these two capabilities, and use the ratio of the next-word probability of the token "`Yes`" and "`No`" as priority values. We refer to these two values as *existing confidence* and *latent confidence*, denoted as c_e and c_l.

The overall priority value for a node is the weighted sum of c_e and c_l. We introduce a weight function $\mathcal{W}(d)$ that is related to a node's depth. When the depth is shallow, indicating minimal zoom and the MLLM might not clearly perceive the cue, assign more weight c_l. As depth increases, shift more weight to c_e. Finally, ranking function $\mathcal{R}$ is introduced to rank nodes by the overall priority value, as shown in Algorithm 2.

3.4 Stopping Criterion

Zoom Eye exits the search process when the MLLM provides feedback that the current view

is sufficient to answer the provided question, denoted as q_s. Specifically, we query the MLLM with a prompt $\mathrm{p}_a(q_s)$ (e.g., "`Could you answer` q_s `now?`") and use the same method as described in §3.3 to quantify the positive feedback. We refer to it as *answering confidence*, denoted as c_a. When c_a exceeds a predefined threshold τ, the search terminates. The implementation of $\mathcal{S}$ is shown in Algorithm 2.

3.5 Overall Search Algorithm

With the above notations in place, we now describe how Zoom Eye works for a given image-question pair ($\mathbf{I}$, q). The complete algorithm workflow is shown in Appendix D.4.

Generating visual cues to guide the search. Before search, the MLLM has to predefine the visual cues essential for addressing q, enabling a targeted and guided search based on these cues. We utilize the in-context capability from the LLM base of the MLLM, using a sequence of contextual examples as prefixes to generate visual cues. Ultimately, the MLLM produces k visual cues $\{o_1, \ldots, o_k\}$ pertinent to q. Each o_i ($i \in \{1, \ldots, k\}$) can be categorized into two types: (type 1) those requiring a search for a single instance, and (type 2) those requiring identification of all instances in the image.

	Question	Visual cues	Type
1	What is the color of the dog?	*dog*	type 1
2	What is the relative position of the dog to the cat?	*dog*, *cat*	type 1, type 1
3	How many dogs in the image?	*all dogs*	type 2

Table 1: Examples of visual cues and their types.

Searching for cues. For each cue o_i ($i \in \{1, \ldots, k\}$), Zoom Eye explores the image tree to capture pertinent visual information. When searching for type 1 cues, the search is guided with $\mathcal{R}$ and concludes as soon as it meets $\mathcal{S}$, then the current node is recorded in a list L . For a single type 1 clue, as shown in line 1 of Table 1, the applied q_s for$\mathcal{S}$ is the input question q. If multiple type 1 clues are generated as in line 2 of Table 1, we introduce a decomposed question template $\mathrm{p}_{dq}(o_i)$ such as "`what is the location of the` $\{o_i\}$`?`" specific to each cue. In this case, the applied q_s of o_i is $\mathrm{p}_{dq}(o_i)$. If a type 2 cue is generated, as shown in line 3 of Table 1, $\mathcal{S}$ is not applied, and we search the whole tree to add all nodes with sufficient *existing confidence* to L.

Answering the question using the searched cues. Given the searched nodes $L = \{n_1^*, \ldots, n_K^*\}$, the MLLM formulates a response to the input question q by synthesizing information of these nodes. Denoting $\mathbf{b}_i^* = (x_{1,i}^*, y_{1,i}^*, x_{2,i}^*, y_{2,i}^*)$ as the bounding-box of n_i^* ($i \in \{1, \ldots, K\}$), we union the bounding-box coordinates of all nodes in L to create a union bounding-box $\mathbf{b}^* = (\min_i x_{1,i}^*, \min_i y_{1,i}^*, \max_i x_{2,i}^*, \max_i y_{2,i}^*)$. For the two distinct image input methods, we apply Eq. 1 to feed the focused region $\mathbf{b}^*$ along with q into models and derive the final response.

4 Experiments

4.1 Implementation Details

Local input. We select LLaVA-v1.5-7B (Liu et al., 2024a) as the base MLLM, with the naive image processing. We set τ at 0.8 and define $\mathcal{W}$ as $\frac{1-b}{D^2} \times d^2 + b$, where D denotes the depth of the image tree, d is the depth of the visited node during the search, and b is a bias value, set here at 0.2.

Global + Local input. We select Qwen2.5VL-3B (Bai et al., 2025), LLaVA-ov(oneVision)-7B (Li et al., 2024), and InternVL2.5-8B (Chen et al., 2024a) as our MLLMs, with the AnyRes image processing. For LLaVA-ov and InternVL, we define the maximum AnyRes block as 12, and for QwenVL, we set the max pixels as 12, 845, 056. We set τ at 0.6 and define $\mathcal{W}$ similarly to the above, except with b of 0.6.

For both input implementation, we set the maximum search depth at 2 when searching for type 2 cues to save costs. Additionally, the decomposed question template $\mathrm{p}_{dq}(o_i)$ is assigned as "`What is the appearance of the` $\{o_i\}$`?`". More details are described in Appendix D.

4.2 Results on High-Resolution Benchmark

Evaluated benchmark. We evaluate Zoom Eye on two meticulously curated high-resolution benchmarks. The first, **V* Bench** (Wu and Xie, 2024), with an average resolution of 2246x1582, features sub-tasks in attribute recognition and spatial reasoning. The second, **HR-Bench 8K** (Wang et al., 2024) boasts average resolution of 7680, which consists of two sub-tasks: Fine-grained Single-instance Perception (FSP) and Fine-grained Cross-instance Perception (FCP). The 8K images are cropped around the objects in question to produce **HR-Bench 4K**. Both benchmarks are comprised of rich visual elements and required detailed perception to accurately respond. More results are displayed in Ta-

Model	V^* Bench			HR-Bench 4K			HR-Bench 8K		
	Attr	Spatial	Overall	FSP	FCP	Overall	FSP	FCP	Overall
Open-source MLLMs									
minigptv2-7B (Chen et al., 2023a)	-	-	-	25.75	25.25	25.50	26.0	26.25	26.13
LLaVA-v1.6-7B (Liu et al., 2024b)	60.87	63.16	61.78	49.0	46.75	47.88	37.25	44.25	40.75
LLaVA-v1.6-13B (Liu et al., 2024b)	60.0	64.47	61.78	49.75	41.25	45.50	38.0	38.25	38.13
Yi-VL-34B (AI et al., 2024)	-	-	-	46.0	42.75	44.38	39.50	38.50	39.0
LLaVA-HR-X-7B (Luo et al., 2024)	51.30	64.47	56.54	57.75	46.25	52.0	42.0	41.25	41.63
Closed-source MLLMs									
QWen-VL-max (Bai et al., 2023)	-	-	-	65.0	52.0	58.50	54.0	51.0	52.50
GPT4o (Achiam et al., 2023)	-	-	66.0	70.0	48.0	59.0	62.0	49.0	55.5
Baseline and Local Input Zoom Eye									
LLaVA-v1.5-7B (Liu et al., 2024a)	43.47	56.57	48.68	38.5	33.75	36.13	33.0	31.25	32.13
LLaVA-v1.5-7B w/ Zoom Eye	83.45	82.89	83.25	67.75	38.75	53.25	65.50	36.0	50.75
Δ	+40.48	+26.32	+34.57	+29.25	+5.0	+17.12	+32.50	+4.75	+18.62
Baseline and Global+Local Input Zoom Eye									
Qwen2.5VL-3B (Bai et al., 2025)	80.87	71.05	76.96	82.75	49.0	65.88	80.5	45.25	62.88
Qwen2.5VL-3B w/ Zoom Eye	88.70	**89.47**	89.01	86.75	53.50	70.13	84.75	52.0	68.38
Δ	+7.83	+18.42	+12.05	+4.0	+4.50	+4.25	+4.25	+6.75	+5.50
LLaVA-ov-7B (Li et al., 2024)	75.65	75.0	75.39	72.0	54.0	63.0	67.25	52.25	59.75
LLaVA-ov-7B w/ Zoom Eye	**93.91**	85.53	**90.58**	84.25	55.0	69.63	88.5	50.0	69.25
Δ	+18.26	+10.53	+14.19	+12.25	+1.0	+6.63	+21.25	-2.25	+10.0
InternVL2.5-8B (Chen et al., 2024a)	67.83	71.05	69.11	75.75	56.25	66.0	61.5	53.25	57.38
InternVL2.5-8B w/ Zoom Eye	86.09	82.89	84.82	**88.75**	**61.50**	**75.13**	**89.75**	**57.5**	**73.63**
Δ	+18.26	+11.84	+15.71	+13.0	+5.25	+9.13	+28.25	+4.25	+16.25

Table 2: Results of different models on high-resolution benchmarks. FSP: Fine-grained Single-instance Perception; FCP: Finegrained Cross-instance Perception. More results are displayed in Table 5.

Method	MO					AD			RS	
	Calculate	Intention	Property	Orientation	Color†	Intention†	Attention	Motion†	Count	Position
LLaVA-ov-7B	36.33	27.55	55.0	14.94	34.19	37.32	71.89	30.61	32.95	61.40
w/ Zoom Eye	38.67	38.78	60.0	14.62	47.09	38.56	68.66	42.71	35.56	48.45
Δ	+2.34	+11.23	+5.0	-0.32	+12.90	+1.24	-3.23	+12.10	+2.61	-12.95

Table 3: Performance comparison on MME-RealWorld benchmark. This benchmark comprises numerous sub-tasks, and we only list those that exhibit obvious performance changes of Zoom Eye against the baseline. MO (Monitoring), AD (Autonomous Driving), and RS (Remote Sensing) are data categories within this benchmark. †This result is an average derived from multiple similar sub-tasks (e.g., Color is the average of Vehicle Color and Person Color).

ble 5.

Main results. As shown in Table 2, all evaluated models exhibit significant performance gains after incorporating Zoom Eye, highlighting its model-agnostic applicability. For instance, LLaVA-ov-7B achieves performance improvements of 14.19%, 6.63%, and 10.00% on V^* Bench, HR-Bench 4K, and HR-Bench 8K, respectively. In conjunction with the case studies presented in Figure 5, these results demonstrate that vision-level reasoning enables MLLMs to more effectively capture fine-grained and task-relevant visual information in complex scenes, thereby enhancing their overall visual understanding capabilities.

4.3 Results on Real-World Benchmark

Evaluated benchmark. We further evaluate Zoom Eye on MME-RealWorld (Zhang et al., 2024), a manually annotated benchmark tailored for real-world applications, featuring an average resolution of 2000×1500. It includes 5 data categories and 43 sub-class tasks. Due to the page limit, we report on only 13 sub-tasks that show significant performance changes with Zoom Eye. These sub-tasks span 3 data categories, with similar types merged (e.g., Vehicle Color and Person Color into Color) to present average scores. Detailed results are provided in Appendix C.

Results. As shown in Table 3, Zoom Eye improves the performance of LLaVA-ov-7B on most sub-tasks, especially on MO/Intention (+11.23%), MO/Color (+12.9%), and AD/Motion (+12.1%). However, we also notice that the model's performance with Zoom Eye decline on some sub-tasks. We selecte one error example each from MO/Orientation and RS/Position and display them in Figure 5. For MO/Orientation, the low direct response scores for LLaVA-ov, as seen in the Table 3, along with error example in the figure, suggest a probable deficiency of orientation data during training, negatively impacting model performance in this aspect. For RS/Position, despite Zoom Eye locates the target, the final response was incorrect, suggesting the model struggles to link positional relationships between the full image and sub-images, resulting in a marked decline in performance on this sub-task.

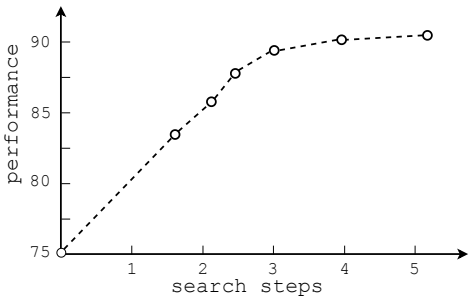

Figure 4: The relationship between the number of search steps and the performance of the MLLM. The experimental statistics are derived from LLaVA-ov-7B's results on V^* Bench.

These error examples reveal the model's deficiencies, by which we will guide the direction of improvements in the model's capabilities in our future work.

4.4 Ablation Studies

4.4.1 Vision-level test-time scaling

We progressively reduce the answering confidence threshold τ and analyze the relationship between the number of search steps and the performance of the MLLM, as illustrated in Figure 4.

From the figure, it can be seen that as the number of search steps increases, the model performance improves and eventually stabilizes. This behavior is analogous to the test-time scaling in text-level reasoning, where the accuracy of the final answer improves with more CoT tokens being explored. This finding could be viewed as a form of **vision-level test-time scaling**, where exploring more detailed zoomed information instead of the static image could enhance the ability of MLLM to generate more accurate responses.

When deploying Zoom Eye in real-world scenarios, we can adjust the confidence threshold or the maximum number of search steps based on specific needs to achieve the best trade-off between performance and efficiency.

4.4.2 Does the Zoom operation contribute to the improvement of the MLLM?

By comparing the answer accuracy of MLLM when Zoom is successful versus when it fails, we investigate the contribution of the Zoom operation to the model. As shown in Table 4, the accuracy sees a remarkable improvement (from 54.55% to 93.45%) when Zoom is successfully applied. This substantial gain highlights the critical role of the Zoom operation. By effectively refining the model's focus on relevant visual details, it contributes to more accurate and reliable responses, reinforcing its importance as a key mechanism for optimizing visual understanding.

Used MLLM	*Zoom Successfully*	*Performance*
LLaVA-ov-7B	✓	93.45
LLaVA-ov-7B	✗	54.55

Table 4: Comparison of MLLM performance conditioned on whether zoom is successful. A zoom is considered successful when the searched box covers at least 50% of the target object. The experimental statistics are derived from V^* Bench.

4.5 Case Study

We visualize some cases in Figure 5, along with error examples mentioned in §4.3. We present cases for single type 1 cue, multiple type 1 cues, and type 2 cue, which is corresponding to the examples in Table 1. From the figure, it can be observed that Zoom Eye accurately seeks out cues, enabling the MLLM to focus on the crucial visual information and respond to queries precisely.

5 Related Work

Multimodal LLMs. Since the advent of large language models (LLMs), they have achieved success across various linguistic applications, such as in-context learning (Dong et al., 2022; Zhang et al., 2022) and retrieval augmented generation (Liu et al., 2024d; Zhao et al., 2024b,c), which facilitated the emergence of Multimodal LLMs, with pioneering works including (Alayrac et al., 2022; Li et al., 2023; Koh et al., 2023). Following these, LLaVA (Liu et al., 2024c) employed GPT-4 (Achiam et al., 2023) to develop training data, inspiring a series of works focused on visual instruction data (Liu et al., 2024a; Dai et al., 2023; Chen et al., 2023b). Since these models utilize pretrained vision encoders (Radford et al., 2021; Zhai et al., 2023) to process image information, the resolution that MLLMs can handle is limited by the input resolution of these encoders. To address this limitation, AnyRes was developed to flexibly manage images of varying resolutions (Liu et al., 2024b; Chen et al., 2024b). Additionally, there are efforts focused on utilizing high-resolution encoders to accommodate high-resolution images (Lu et al., 2024; Wei et al., 2025). However, despite these efforts, the perception of the image by the MLLM remains as the original image itself. We hope to enable MLLMs to explore the varying hierarchical features of images to capture key information.

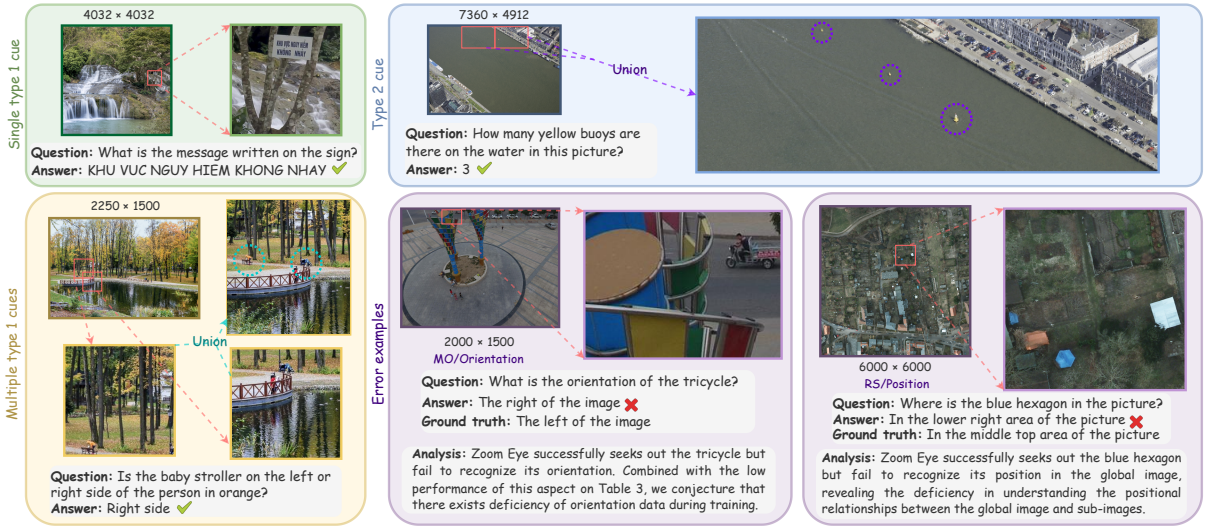

Figure 5: Examples of Zoom Eye. The resolution of the image is displayed. Red rectangles are patches searched by Zoom Eye.

Tree-based search. Tree-based search algorithms have been applied in text-only LLM reasoning and have demonstrated superior performance. Early works such as (Wei et al., 2022; Wang et al., 2022) relied on chain reasoning, a method susceptible to errors in one step propagating through subsequent steps. Consequently, ToT (Yao et al., 2024b) proposed a tree-based reasoning method that leverages the expansiveness of tree structures to widen the reasoning space. Simultaneously, several similar studies were also introduced, which define a decomposed question step as a node and utilize beam search (Xie et al., 2023) and Monte-Carlo Tree Search (Hao et al., 2023) to uncover optimal solutions. Subsequently, TS-LLM (Feng et al., 2023) utilized reinforcement learning to increase search depth, further enhancing reasoning performance. In our work, we conceptualize an image as a tree to search for crucial visual information using a specific algorithm. A close-related work is V^* (Wu and Xie, 2024), and we describe the detailed comparison with it in Appendix B.

6 Limitations

Although Zoom Eye offers several advantages, such as strong interpretability, model-agnostic, and training-free, it also comes with certain limitations. First, the current search procedure relies on heuristic strategies, including manually defined ranking functions and stopping criteria. While these designs are effective in many settings, they may not generalize optimally across all image types or task conditions. Second, the image is partitioned into fixed-size patches to construct the hierarchical tree structure, which may not align well with the semantic regions of the image. As a result, some visual cues may be fragmented or overlooked during traversal. Lastly, Zoom Eye is primarily tailored for natural images with spatially distributed visual elements. It is less applicable to document understanding tasks, where layout, reading order, and structured information (e.g., tables, forms) are central. Addressing these challenges—such as by integrating learnable search strategies or adaptive patch partitioning—will be an important direction for future work.

7 Conclusion

To address the limitations of text-level visual reasoning, we propose Zoom Eye, a type of vision-level reasoning method, a tree search algorithm designed to navigate the hierarchical and visual nature of images to capture detailed crucial information. Through prompts guiding MLLMs, we develop a ranking function and stopping criterion for Zoom Eye, which steers models to efficiently search along the image tree, seek out pertinent information, and accurately respond to related queries. Experiments show the broad-applicability and effectiveness of Zoom Eye, which substantially improves MLLMs' performance. Notably, Zoom Eye exhibits a test-time scaling phenomenon analogous to that observed in text-level reasoning. Meanwhile, through the analysis of failure cases, we identify several inherent limitations in current MLLMs' visual reasoning capabilities, which we aim to address in future work.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

01. AI, :, Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, Kaidong Yu, Peng Liu, Qiang Liu, Shawn Yue, Senbin Yang, Shiming Yang, Tao Yu, and 13 others. 2024. Yi: Open foundation models by 01.ai. *Preprint*, arXiv:2403.04652.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, and 1 others. 2022. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*, 1(2):3.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Jun Chen, Deyao Zhu, Xiaoqian Shen, Xiang Li, Zechun Liu, Pengchuan Zhang, Raghuraman Krishnamoorthi, Vikas Chandra, Yunyang Xiong, and Mohamed Elhoseiny. 2023a. Minigpt-v2: large language model as a unified interface for vision-language multi-task learning. *arXiv preprint arXiv:2310.09478*.

Lin Chen, Jisong Li, Xiaoyi Dong, Pan Zhang, Conghui He, Jiaqi Wang, Feng Zhao, and Dahua Lin. 2023b. Sharegpt4v: Improving large multimodal models with better captions. *arXiv preprint arXiv:2311.12793*.

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, and 1 others. 2024a. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*.

Zhe Chen, Weiyun Wang, Hao Tian, Shenglong Ye, Zhangwei Gao, Erfei Cui, Wenwen Tong, Kongzhi Hu, Jiapeng Luo, Zheng Ma, and 1 others. 2024b. How far are we to gpt-4v? closing the gap to commercial multimodal models with open-source suites. *arXiv preprint arXiv:2404.16821*.

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. 2023. Instructblip: Towards general-purpose vision-language models with instruction tuning. *Preprint*, arXiv:2305.06500.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, and 1 others. 2022. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*.

Yuhao Dong, Zuyan Liu, Hai-Long Sun, Jingkang Yang, Winston Hu, Yongming Rao, and Ziwei Liu. 2024. Insight-v: Exploring long-chain visual reasoning with multimodal large language models. *arXiv preprint arXiv:2411.14432*.

Xidong Feng, Ziyu Wan, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. 2023. Alphazero-like tree-search can guide large language model decoding and training. *arXiv preprint arXiv:2309.17179*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Jing Yu Koh, Ruslan Salakhutdinov, and Daniel Fried. 2023. Grounding language models to images for multimodal inputs and outputs. In *International Conference on Machine Learning*, pages 17283–17300. PMLR.

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

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024a. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26296–26306.

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

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024c. Visual instruction tuning. *Advances in neural information processing systems*, 36.

Jingyu Liu, Jiaen Lin, and Yong Liu. 2024d. How much can rag help the reasoning of llm? *arXiv preprint arXiv:2410.02338*.

Haoyu Lu, Wen Liu, Bo Zhang, Bingxuan Wang, Kai Dong, Bo Liu, Jingxiang Sun, Tongzheng Ren, Zhuoshu Li, Hao Yang, and 1 others. 2024. Deepseek-vl: towards real-world vision-language understanding. *arXiv preprint arXiv:2403.05525*.

Gen Luo, Yiyi Zhou, Yuxin Zhang, Xiawu Zheng, Xiaoshuai Sun, and Rongrong Ji. 2024. Feast your eyes: Mixture-of-resolution adaptation for multimodal large language models. *arXiv preprint arXiv:2403.03003*.

Fanqing Meng, Lingxiao Du, Zongkai Liu, Zhixiang Zhou, Quanfeng Lu, Daocheng Fu, Botian Shi, Wenhai Wang, Junjun He, Kaipeng Zhang, and 1 others. 2025. Mm-eureka: Exploring visual aha moment with rule-based large-scale reinforcement learning. *arXiv preprint arXiv:2503.07365*.

M Minderer, A Gritsenko, A Stone, M Neumann, D Weissenborn, A Dosovitskiy, A Mahendran, A Arnab, M Dehghani, Z Shen, and 1 others. 2022. Simple open-vocabulary object detection with vision transformers. arxiv 2022. *arXiv preprint arXiv:2205.06230*, 2.

OpenAI. 2025. o3/o4 mini system card. https://openai.com/index/o3-o4-mini-system-card/.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Haozhan Shen, Peng Liu, Jingcheng Li, Chunxin Fang, Yibo Ma, Jiajia Liao, Qiaoli Shen, Zilun Zhang, Kangjia Zhao, Qianqian Zhang, Ruochen Xu, and Tiancheng Zhao. 2025. Vlm-r1: A stable and generalizable r1-style large vision-language model. *arXiv preprint arXiv:2504.07615*.

Quan Sun, Yuxin Fang, Ledell Wu, Xinlong Wang, and Yue Cao. 2023. Eva-clip: Improved training techniques for clip at scale. *arXiv preprint arXiv:2303.15389*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Wenbin Wang, Liang Ding, Minyan Zeng, Xiabin Zhou, Li Shen, Yong Luo, and Dacheng Tao. 2024. Divide, conquer and combine: A training-free framework for high-resolution image perception in multimodal large language models. *arXiv preprint arXiv:2408.15556*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Haoran Wei, Lingyu Kong, Jinyue Chen, Liang Zhao, Zheng Ge, Jinrong Yang, Jianjian Sun, Chunrui Han, and Xiangyu Zhang. 2025. Vary: Scaling up the vision vocabulary for large vision-language model. In *European Conference on Computer Vision*, pages 408–424. Springer.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Penghao Wu and Saining Xie. 2024. V?: Guided visual search as a core mechanism in multimodal llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13084–13094.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, Min-Yen Kan, Junxian He, and Qizhe Xie. 2023. Decomposition enhances reasoning via self-evaluation guided decoding. *Preprint*, arXiv:2305.00633.

Guowei Xu, Peng Jin, Hao Li, Yibing Song, Lichao Sun, and Li Yuan. 2024. Llava-cot: Let vision language models reason step-by-step. *Preprint*, arXiv:2411.10440.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, and 1 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Huanjin Yao, Jiaxing Huang, Wenhao Wu, Jingyi Zhang, Yibo Wang, Shunyu Liu, Yingjie Wang, Yuxin Song, Haocheng Feng, Li Shen, and 1 others. 2024a. Mulberry: Empowering mllm with o1-like reasoning and reflection via collective monte carlo tree search. *arXiv preprint arXiv:2412.18319*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024b. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. 2023. Sigmoid loss for language image pre-training. *Preprint*, arXiv:2303.15343.

Yi-Fan Zhang, Huanyu Zhang, Haochen Tian, Chaoyou Fu, Shuangqing Zhang, Junfei Wu, Feng Li, Kun Wang, Qingsong Wen, Zhang Zhang, and 1 others. 2024. Mme-realworld: Could your multimodal llm challenge high-resolution real-world scenarios that are difficult for humans? *arXiv preprint arXiv:2408.13257*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*.

Tiancheng Zhao, Qianqian Zhang, Kyusong Lee, Peng Liu, Lu Zhang, Chunxin Fang, Jiajia Liao, Kelei Jiang, Yibo Ma, and Ruochen Xu. 2024a. Omchat: A recipe to train multimodal language models with strong long context and video understanding. *arXiv preprint arXiv:2407.04923*.

Xinping Zhao, Dongfang Li, Yan Zhong, Boren Hu, Yibin Chen, Baotian Hu, and Min Zhang. 2024b. Seer: Self-aligned evidence extraction for retrieval-augmented generation. *arXiv preprint arXiv:2410.11315*.

Xinping Zhao, Yan Zhong, Zetian Sun, Xinshuo Hu, Zhenyu Liu, Dongfang Li, Baotian Hu, and Min Zhang. 2024c. Funnelrag: A coarse-to-fine progressive retrieval paradigm for rag. *arXiv preprint arXiv:2410.10293*.

Xinyu Zhu, Junjie Wang, Lin Zhang, Yuxiang Zhang, Yongfeng Huang, Jiaxing Zhang, Yujiu Yang, and 1 others. 2023. Solving math word problems via cooperative reasoning induced language models. In *The 61st Annual Meeting Of The Association For Computational Linguistics*.

A Results of More MLLMs on High-Resolution Benchmark

We present the results of additional MLLMs on high-resolution benchmarks in Table 5, including models of smaller or larger scale. Consistent with the findings in the main paper, all evaluated models exhibit improved performance after being adapted to Zoom Eye, further demonstrating the effectiveness of vision-level reasoning in handling complex visual scenarios.

B Compared with Other HR Processing Methods

Compared method. We choose two high-resolution processing methods to compare—V^* (Wu and Xie, 2024) (published in *CVPR 2024*) and DC^2 (Wang et al., 2024) (published in *AAAI 2025*). V^* is a LLM-guided search pipeline for MLLMs. To match the input resolution of the V^* model, we specifically trained a 224px version of the LLaVA-v1.5 model for a fair comparison. Apart from using CLIP-224 (Radford et al., 2021) as the vision encoder, all other settings were identical to those of LLaVA-v1.5.

DC^2 (**D**ivide, **C**onquer, and **C**ombine) is a framework that supplements visual information using text for high-resolution images understanding. Like our approach, it recursively splits an image into patches. The MLLM then generates textual descriptions for each leaf patch. These descriptions are then relayed to the parent nodes, which create combined descriptions by synthesizing the contents from their child nodes with their own. This process continues up to the root node.

B.1 Zoom Eye vs. V*.

From Table 6, it is evident that compared to V^*, our method offers several advantages: (1) The V^* pipeline requires specifically targeted training data, making zero-shot searches impossible, whereas our method utilizes the native capabilities of MLLMs, allowing adaptation to any MLLM without additional training; (2) V^*'s search process necessitates the integration of another specially trained MLLM to guide the search, along with an extra high-resolution image encoder (Minderer et al., 2022)(768px), while our approach operates at the native resolution of MLLMs and conducts searches independently; (3) Our method demonstrates superior performance.

Model	V^* Bench			HR-Bench 4K			HR-Bench 8K		
	Attr	Spatial	Overall	FSP	FCP	Overall	FSP	FCP	Overall
Baseline and Local Input Zoom Eye									
LLaVA-v1.5-13B (Liu et al., 2024a)	41.74	55.26	47.12	45.25	41.25	43.25	37.50	38.0	37.75
LLaVA-v1.5-13B w/ Zoom Eye	87.83	81.58	85.34	73.0	43.25	58.13	67.25	45.50	56.38
Δ	+46.09	+26.32	+38.22	+27.75	+2.00	+14.88	+29.75	+7.50	+18.63
Baseline and Global+Local Input Zoom Eye									
LLaVA-ov-0.5B (Li et al., 2024)	63.48	64.47	63.87	63.50	39.50	51.50	47.25	38.25	42.75
LLaVA-ov-0.5B w/ Zoom Eye	85.22	73.68	80.62	75.50	39.75	57.63	68.50	38.25	53.38
Δ	+21.74	+9.21	+16.75	+12.00	+0.25	+6.13	+21.25	+0.00	+10.63
InternVL2.5-4B (Chen et al., 2024a)	69.57	71.05	70.16	77.50	53.75	65.63	63.00	49.25	56.13
InternVL2.5-4B w/ Zoom Eye	85.22	77.63	82.20	81.25	56.75	69.00	80.00	52.25	66.13
Δ	+15.65	+6.58	+12.04	+3.75	+3.00	+3.37	+17.00	+3.00	+10.00
InternVL2.5-26B (Chen et al., 2024a)	73.91	72.37	73.30	82.00	66.25	74.13	73.00	61.75	67.38
InternVL2.5-26B w/ Zoom Eye	91.30	86.84	89.53	89.75	68.25	79.00	89.25	63.00	76.13
Δ	+17.39	+14.47	+16.23	+7.75	+2.00	+4.87	+16.25	+1.25	+8.75

Table 5: Results of more models on high-resolution benchmarks.

Method	Input Res.	Search Res.	Zero shot	Indep. search	V^* Bench	HR Bench
V^* search(Wu and Xie, 2024)	224	768	✗	✗	75.39	37.81
Zoom Eye	224	224	✓	✓	**81.58**	**47.63**

Table 6: Performance comparison between Zoom Eye and V^* Search. Input Res.: The input resolution of the model generating the final response; Search Res.: The resolution required during the search process; Zero shot: Whether the method could be adapted for models without the need for specialized additional training; Indep. search: Whether the method could be applied directly to an MLLM instead of requiring an additional search model.

Method	V^* Bench	HR-Bench 8K
LLaVA-v1.5-7B	48.7	32.1
w/ DC^2 (Wang et al., 2024)	57.3	39.5
w/ Zoom Eye	**83.3**	**50.8**
LLaVA-v1.5-13B	47.1	37.8
w/ DC^2	57.3	40.5
w/ Zoom Eye	**85.3**	**56.4**

Table 7: Performance comparison between Zoom Eye and DC^2.

B.2 Zoom Eye vs. DC^2.

Our approach differs from DC^2 in two key ways: (1) DC^2 uses textual modalities to supplement the missing visual information at high resolutions, whereas Zoom Eye employs simulated zooming operations, allowing the MLLM to actively discover missing visual details and provide more pertinent information for multimodal tasks; (2) DC^2 is question-agnostic, generating descriptions consistently across different questions which may lead to unfocused textual content. In contrast, Zoom Eye is question-driven in its visual cues searching, yielding more precise visual information that is instrumental in answering the input question. Table 7 shows that Zoom Eye performs better than DC^2 using the same MLLM.

C Complete Results on MME-RealWorld Benchmark

We provide the complete results of Zoom Eye on MME-RealWorld Benchmark (Zhang et al., 2024), as show in Table 8. This benchmark includes 5 data categories: Monitoring (MO), Autonomous Driving (AD), OCR, emote Sensing (RS), and Diagram and Table (TD). Since Zoom Eye is not applicable to the TD task, we do not conduct tests on it. It could be observed that Zoom Eye improves the performance of LLaVA-ov-7B across most subtasks, with particularly significant improvements in certain tasks. For instance, it achieves a 20.22% improvement in the $\text{Person}_{\text{color}}$ task, a 29.11% improvement in the $\text{Motion}_{\text{vehicle}}$ task, and a 12.93% improvement in the $\text{Visual}_{\text{trafficsignal}}$ task, demonstrating the effectiveness of Zoom Eye. However, performance declines were observed in some subtasks when using Zoom Eye. We have analyzed these cases in the main paper, revealing certain limitations of the employed MLLM. Addressing these issues will be a focus of our future work.

D Implementation Details

Due to the page limit of the main paper, we provide more implementation details here. In §D.1 and §D.2, we detail the implementation of Local Input and Global+Local Input, respectively. §D.3 describes the implementations common to both. Finally, based on the introductions in the first three subsections, we present the complete algorithm workflow of Zoom Eye in §D.4.

D.1 Local Input

We select LLaVA-v1.5-7B (Liu et al., 2024a) and 13B as our MLLMs, with the vision encoder's input

Task		LLaVA$_{ov}$-7B	+ZoomEye	Δ ↑
MO	Calculate	36.33	38.67	+2.34
	Intention	27.55	38.78	+11.23
	Property	55.0	60.0	+5.0
	Vehicle$_{counting}$	59.89	61.14	+1.25
	Person$_{counting}$	61.35	61.87	+0.52
	Vehicle$_{location}$	33.82	33.82	-
	Vehicle$_{orientation}$	19.35	18.71	-0.64
	Vehicle$_{color}$	43.65	49.24	+5.59
	Person$_{color}$	24.72	44.94	+20.22
	Person$_{orientation}$	10.53	10.53	-
AD	Intention$_{ego}$	28.62	28.95	+0.33
	Intention$_{pedestrian}$	52.43	53.40	+0.97
	Intention$_{vehicle}$	30.92	33.33	+2.41
	Interaction$_{other2other}$	12.94	13.43	+0.49
	Attention$_{trafficsignal}$	71.89	68.66	-3.23
	Interaction$_{ego2pedestrain}$	27.36	28.30	+0.94
	Interaction$_{ego2trafficsignal}$	22.86	25.71	+2.85
	Interaction$_{ego2vehicle}$	20.79	19.80	-0.99
	Objects$_{identify}$	64.40	64.85	+0.45
	Motion$_{vehicle}$	23.42	52.53	+29.11
	Motion$_{multivehicles}$	34.26	34.75	+0.49
	Visual$_{trafficsignal}$	60.20	73.13	+12.93
	Motion$_{pedestrain}$	34.15	40.85	+6.70
	Object$_{count}$	37.92	39.86	+1.94
	Motion$_{multipedestrians}$	31.24	31.64	+0.40
OCR	Scene understanding	64.80	64.80	-
	Character identification	57.60	56.40	-1.20
	Adver & product	76.64	78.37	+1.73
	Book map poster	77.17	75.24	-1.93
	License	80.16	82.39	+2.23
	Phone & address	77.82	81.28	+3.46
	Text recog	74.87	77.13	+2.26
RS	Color	59.60	60.56	+0.96
	Count	32.95	35.56	+2.61
	Position	61.40	48.45	-12.95

Table 8: Performance comparison between Zoom Eye and the baseline model on MME-RealWorld benchmark. MO (Monitoring), AD (Autonomous Driving), OCR and RS (Remote Sensing) are data categories within this benchmark.

resolution as 336px and naive processing. We set the threshold of the stopping criterion at $\tau = 0.8$ and define the weighted function as $\mathcal{W} = \frac{1-b}{D^2} \times d^2 + b$, where D denotes the depth of the image tree, d is the depth of the visited node during the search, and b is a bias value, set here at 0.2. The prompt templates for calculating *existing confidence*, *latent confidence*, and *answering confidence* (please refer to §3.3 and §3.4 for the discussion on these three confidence values) are set as:

Prompt Templates of Local Input

- $p_e(o)$: *<local patch> Is there a {o} in the image? Answer Yes or No.*
- $p_l(o)$: *<local patch> According to your common sense knowledge and the content of the image, is it possible to find a {o} in the image? Answer Yes or No and tell the reason.*
- $p_a(q)$: *<local patch> Question: {q} \nCould you answer the question based on the the available visual information? Answer Yes or No.*

where the o and q are the input visual cue and question, which could be referred to §3.5 .

As mentioned in §3.5, the final visual input uses the union of all searched patches. However, when multiple distant patches are combined, they may form a large image. For MLLMs using naive resize processing, information can still be lost during downsampling. Therefore, for the Local Input Zoom Eye with naive resize processing, when the area of b^* is relatively large (with the longer side exceeding 1000px), we skip the Union operation. Instead, we paste the searched patches onto a blank image according to their relative positions in the original image, and then feed it to the MLLMs. An example is shown in Figure 6.

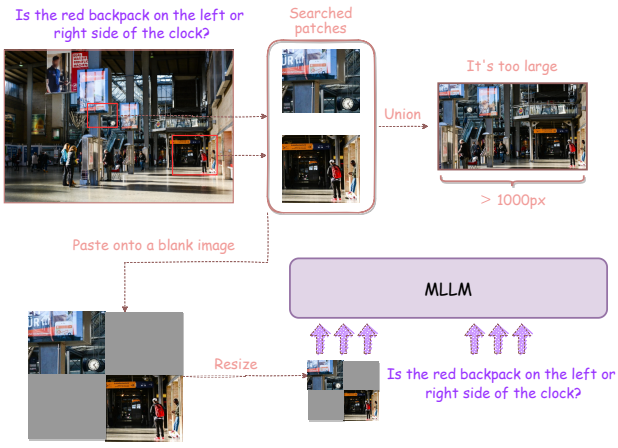

Figure 6: If the area of the union bounding box is too large, we paste the searched patches onto a blank image according to their relative positions in the original image, and then feed it to the MLLMs.**It is notable that this operation is only applied to Local Input, while for Local+Global, we consistently provide the MLLMs with the full union patch as input.**

D.2 Global + Local Input

We select LLaVA-ov(oneVision)-0.5B (Li et al., 2024) and 7B as our MLLMs, with the vision encoder's input resolution as 384px and AnyRes processing. We define the maximum AnyRes block as 12, set τ at 0.6 and define $\mathcal{W}$ as $\frac{1-b}{D^2} \times d^2 + b$, where D denotes the depth of the image tree, d is the depth of the visited node during the search, and b is a bias value, set here at 0.6. The prompt templates for calculating *existing confidence*, *latent confidence*, and *answering confidence* are set as:

Prompt Templates of Global + Local Input

- $p_e(o)$: *<global image><local patch> Is there a $\{o\}$ in the zoomed-in view? Answer Yes or No.*
- $p_l(o)$: *<global image><local patch> According to your common sense knowledge and the content of the zoomed-in view, along with its location in the image, is it possible to find a $\{o\}$ by further zooming in the current view? Answer Yes or No and tell the reason.*
- $p_a(q)$: *<global image><local patch> Question: $\{q\}$ \nCould you answer the question based on the the available visual information? Answer Yes or No.*

D.3 Additional Settings

For both input implementation, we set the maximum search depth at 2 when searching for type 2 cues to save costs. In §3.5, we state that we search the whole tree to add all nodes with sufficient *existing confidence* to L if type 2 cue is generated. Thus, we introduce an additional threshold τ_2 for this condition, which is set at 0.8 for both implementation. The decomposed question template $\mathrm{p}_{dq}(o_i)$ is assigned as "`What is the appearance of the` $\{o_i\}$`?`". For type 1 search, a key aspect is determining the value of τ. If it is set too low, an incorrect patch, which probably lead to erroneous guidance for MLLMs, may be selected. Conversely, setting τ too high, surpassing the c_a values of all nodes in the tree, would compel MLLMs to search the entire tree unnecessarily, thus wasting time. Therefore, we adopt a strategy where τ is progressively reduced as the number of search steps increases. Specifically, if the number of search steps exceeds the step threshold C, we reduce the value of τ by 0.1. This reduction occurs every δ steps, until the c_a value of a node having been visited surpasses τ or τ falls below a predefined minimum limit τ_{min}. For both implementation, we set δ at 2, τ_{min} at 0, and C as $D \times 3$. Finally, the in-context examples we utilized to generate visual cues are denote as $(q^{(1)}, \mathbf{o}^{(1)}, \ldots, q^{(m)}, \mathbf{o}^{(m)})$ and are presented at the end of this document.

D.4 Complete Algorithm Workflow

With the aforementioned notation and description in place, we provide the complete algorithm workflow in Algorithm 3, where the Zoom Eye search method is shown in Algorithm 4.

Algorithm 3 Complete Algorithm Workflow of Zoom Eye

Require: Multimodal LLM Φ_θ, input question-image pair ($\mathbf{I}$, q), decomposed question template p_{dq}, in-context examples $(q^{(1)}, \mathbf{o}^{(1)}, \ldots, q^{(m)}, \mathbf{o}^{(m)}, q)$

$\{o_1, \ldots, o_k\} \leftarrow \Phi_\theta$.generate$(q^{(1)}, \mathbf{o}^{(1)}, \ldots, q^{(m)}, \mathbf{o}^{(m)}, q)$
Initialize L as the empty list
Build $\mathbf{I}$ as a tree T
for $i = 1, \ldots, k$ **do**
  if $k == 1$ **then**
    $q_s \leftarrow q$
  else
    $q_s \leftarrow \mathrm{p}_{dq}(o_i)$
  L.extend(ZOOM EYE(T, o_i, q_s))
$\mathbf{b}^* \leftarrow$ Union bounding-boxes of all nodes in L
$n^* \leftarrow \{\mathbf{I}, \mathbf{b}^*\}$
Final response $\leftarrow \Phi_\theta$.generate$(\mathcal{V}(n^*), q)$

Algorithm 4 Zoom Eye Search

Require: Threshold of type 1 cue and type 2 cue (τ, τ_2), minimum limit τ_{min}, interval δ

function ZOOM EYE(T, o_i, q_s)
  Initialize Q as the empty queue {}
  Q.append(T.root)
  Initialize L_i as the empty list
  search all $\leftarrow o_i$.startswith("all")
  if not search all **then**
    ZOOM EYE TYPE 1(T, Q, L_i, q_s, τ)
  else
    ZOOM EYE TYPE 2(Q, L_i, τ_2)
  return L_i

function ZOOM EYE TYPE 1(T, Q, L_i, q_s, τ)
  import $\mathcal{R}$ and $\mathcal{S}$ from Algorithm 2
  count $\leftarrow 0$
  $C \leftarrow T.depth \times 3$
  Initialize n_m as T.root to record the node with the best c_a
  while Q is not empty **do**
    $n_t \leftarrow Q$.pop()
    N.append(n_t)
    count $\leftarrow$ count + 1
    if count $\geq$ C **then**
      $\tau \leftarrow \tau - 0.1$
      $C \leftarrow C + \delta$
      if $\tau < \tau_{min}$ **then**
        break
    if $\mathcal{S}(n_t, q_s, \tau) ==$ True **then**
      L_i.append(n_t)
      break
    else if $\mathcal{S}(n_m, q_s, \tau) ==$ True **then**
      L_i.append(n_m)
      break
    if $n_t.c_a \geq n_m.c_a$ **then**
      $n_m \leftarrow n_t$
    $s \leftarrow n_t$.children.size
    for $j = 1, \ldots, s$ **do**
      Q.append(n_t.children[j])
    Q.sort($\mathcal{R}(o_i)$)

function ZOOM EYE TYPE 2(Q, L_i, τ_2)
  while Q is not empty **do**
    if n_t.depth ≥ 2 **then**
      break
    $c_e \leftarrow$ calculate the existing confidence of n_t
    if $c_e \geq \tau_2$ **then**
      L_i.append(n_t)
    $s \leftarrow n_t$.children.size
    for $j = 1, \ldots, s$ **do**
      Q.append(n_t.children[j])

In-Context Examples of V^* Bench

User: *Question: What is the color of the boy's bag? If you want to answer the question, which objects' information do you need?*

Assistant: *To answer the question, I need know the location of the boy with a bag so that I can determine the color of the bag. So I need the information about the following objects: boy with a bag.*

User: *Is the yellow car on the left or right side of the white car?*

Assistant: *To answer the question, I need know the location of the yellow car and the white car so that I can determine the positional relationship between the two of them. So I need the information about the following objects: white car and yellow car.*

User: *Tell me the number on the black board.*

Assistant: *To answer the question, I need know the location of the black board so that I can determine the number on it. So I need the information about the following objects: black board*

User: *Is the girl with pink hair on the left or right side of the man with backpack?*

Assistant: *To answer the question, I need know the location of the girl with pink hair and the man with backpack so that I can determine the positional relationship between the two of them. So I need the information about the following objects: girl with pink hair and man with backpack.*

User: *What kind of animal is on the red sign?*

Assistant: *To answer the question, I need know the location of the red sign so that I can determine the kind of animal on it. So I need the information about the following objects: red sign.*

User: *From the information on that advertising board, what is the type of this shop?*

Assistant: *To answer the question, I need know the location of the advertising board so that I can determine the type of the shop. So I need the information about the following objects: advertising board.*

In-Context Examples of HR-Bench

User: *Question: What is the color of the boy's bag? If you want to answer the question, which objects' information do you need?*

Assistant: *To answer the question, I need know the location of the boy with a bag so that I can determine the color of the bag. So I need the information about the following objects: boy with a bag.*

User: *Is the yellow car on the left or right side of the white car?*

Assistant: *To answer the question, I need know the location of the yellow car and the white car so that I can determine the positional relationship between the two of them. So I need the information about the following objects: white car and yellow car.*

User: *Tell me the number on the black board above the dog.*

Assistant: *To answer the question, I need know the location of the black board above the dog so that I can determine the number on it. So I need the information about the following objects: black board above the dog.*

User: *Is the girl with pink hair on the left or right side of the man with backpack?*

Assistant: *To answer the question, I need know the location of the girl with pink hair and the man with backpack so that I can determine the positional relationship between the two of them. So I need the information about the following objects: girl with pink hair and man with backpack.*

User: *What kind of animal is on the red sign?*

Assistant: *To answer the question, I need know the location of the red sign so that I can determine the kind of animal on it. So I need the information about the following objects: red sign.*

User: *How many cars in the image?*

Assistant: *To answer the question, I need know the location of all cars so that I can determine the number of cars. So I need the information about the following objects: all cars.*