A Implementation details

In this section, we provide additional technical details of our work. We begin by describing the hardware setup and report the total GPU hours required to reproduce our main results in Appendix A.1. Next, we explain in detail how the reported metrics are calculated, see Appendix A.2. We then provide the hyperparameters used for FOCUS and the baseline visual cropping methods in Appendix A.3 and Appendix A.4, respectively. Finally, we present further details on the processing scheme of FOCUS in Appendix A.5.

A.1 Hardware specifications

We run all experiments presented in Section 4 and Appendix C on identical hardware, namely compute instances equipped with an *NVIDIA A100 80GB* GPU, an *AMD EPYC 7V13* CPU, and *220 GB* of RAM. FOCUS's software environment includes *CUDA 12.2*, *PyTorch 2.6.0*, and the HuggingFace *transformers* library in version *4.46.0*.

We also provide an estimation of the GPU hours needed to reproduce our results in the entire project. In total, our reported results require approximately 132 GPU hours on the hardware configuration described above (see Table 4). Additionally, we report the GPU hours required to run the experiments that achieve the best-performing variants for each method on LLaVA-1.5 and LLaVA-OneVision.

Table 4: **Estimated GPU hours.**

Model	Best-perf. variant w/ LLaVA-1.5	Best-perf. variant w/ LLaVA-OV	All variants
SEAL	6 [1]	–	6
ViCrop	3	–	6
ZoomEye	5	30	80
FOCUS **(Ours)**	2	9	40
Sum	16	39	132

A.2 Metrics

In this subsection, we provide a detailed description of our performance and efficiency metrics.

Performance metrics The main performance metric is the accuracy of visual cropping methods, which is typically reported as a percentage. We follow the standard evaluation protocol and do not apply any post-processing to the answers generated by the visual cropping methods using the MLLM. This means that if the ground-truth label is `"A"` and the model outputs `"The answer is A"` or `"(A)"`, we do not extract or normalize the answer to match the label. As a result, such responses are counted as incorrect. However, this type of mismatch is of minor importance in our experiments: across all models and three datasets (V*Bench, HRBench-4K, HRBench-8K), we did not observe any irregular or non-standard response formats. We compute the average accuracy as the unweighted mean over all N samples in a dataset, i.e., as $\sum_{i=1}^{N}(\hat{y}_i = y_i)/N$, where $\hat{y}_i$ is the predicted answer and y_i is the ground-truth label for the i-th sample.

Efficiency metrics An important metric is the number of Forward Passes (FPs) required for the visual search. Note that we exclude the FPs needed for the final VQAs prediction, as different inference schemes can generate different forward passes, see Appendix C.3. We compute this by tracking how often the `generate`, `forward`, or `__call__` methods of the respective MLLM are invoked per question. For methods that use multiple MLLMs (e.g., SEAL), we report the total number of FPs of all MLLMs.

Another key metric we report is the efficiency improvement of FOCUS relative to the baseline methods. Among all evaluated approaches, FOCUS demonstrates the highest efficiency. To ensure a fair comparison, we consider the top accuracy achieved by FOCUS and the best-performing baseline, and select the lower of the two as the reference accuracy. For both methods, we determine the number of Forward Passes (FPs) required to reach this reference accuracy—either by taking the exact value or interpolating between data points to estimate the FPs. This yields FP_{ours} for FOCUS and FP_{ref} for the best-performing baseline. The relative efficiency improvement is then computed as $\text{FP}_{\text{ref}}/\text{FP}_{\text{ours}}$.

Furthermore, we report two additional metrics to provide a practical comparison of efficiency, i.e., execution time and peak GPU memory usage (see Table 10). Execution time is recorded per sample, and we report the average

[1] SEAL uses customized MLLMs based on LLaVA-7B.

time per question across a dataset. Peak memory is measured using `torch.cuda.max_memory_allocated()` and converted to GB by dividing the result by 1024^3.

A.3 Hyperparameters of FOCUS

This subsection outlines the hyperparameters used in our method, FOCUS. These parameters are applied consistently across all experiments and correspond to the results reported in Section 4. As noted in the main paper, the only parameter we vary is the number of steps, n_{steps}. Most other hyperparameters remain fixed across experiments for both LLaVA-1.5 and LLaVA-OneVision. The exceptions are three parameters: k and s_{dist} (specific to LLaVA-1.5), and s_{max} (specific to LLaVA-OneVision), as detailed in Table 5.

For LLaVA-1.5, we use a smaller value of k when n_{steps} is low to reduce the number of proposed ROIs. Additionally, we increase s_{dist} on HRBench to ensure a broader spatial distribution of the ROIs across the image. In contrast, for LLaVA-OneVision, we set a larger k due to its higher-resolution object relevance map. Moreover, we set $s_{\text{max}} = 9$ for V*Bench and $s_{\text{max}} = 5$ for the other datasets, as V*Bench contains lower-resolution (2K) images.

The hyperparameters used in FOCUS are generally robust and transferable across a wide range of use cases. For users applying our method to new datasets, we recommend adjusting s_{max}, which determines the maximum size of each proposed ROIs, based on both the resolution of the input images and the spatial resolution of the object relevance map. In particular, s_{max} should be chosen so that the corresponding region in the original image spans approximately 1–2$\times$ the base resolution of the vision encoder. For example, if the object relevance map has a spatial resolution of 60×30, and the input image resolution is 7680×3840, then each grid element corresponds to an area of 128×128. Setting $s_{\text{max}} = 5$ yields a maximum crop size of 640×640, which falls within the recommended range of $384 \times 384 - 768 \times 768$ for the SigLIP encoder. Additionally, if one considers migrating our method to another MLLM, we recommend selecting the last $25\% - 60\%$ of the layers.

Table 5: **Hyperparameters of FOCUS.**

Hyper-parameter	Description	LLaVA-1.5	LLaVA-OneVision
k	Number of anchor points	$k = \begin{cases} 15 & \text{if } n_{\text{steps}} < 4 \\ 30 & \text{otherwise} \end{cases}$	$k = 30$
s_{min}	Minimum size of each ROI	$s_{\text{min}} = 3$	$s_{\text{min}} = 3$
s_{max}	Maximum size of each ROI	$s_{\text{max}} = 5$	$s_{\text{max}} = \begin{cases} 9 & \text{for V*Bench} \\ 5 & \text{otherwise} \end{cases}$
s_{dist}	Minimum Euclidean distance between anchor points	$s_{\text{dist}} = \begin{cases} 2 & \text{for V*Bench} \\ 3 & \text{otherwise} \end{cases}$	$s_{\text{dist}} = 2$
l	Start layer of the used MLLM-internal representations	$l = 14$	$l = 21$
L	End layer of the used MLLM-internal representations	$L = 32$	$L = 28$
t_{type2}	Threshold for inclusion of ROIs for type-2 questions	$t_{\text{type2}} = 0.6$	$t_{\text{type2}} = 0.5$
$t_{\text{obj\_dist}}$	Threshold for merging ROIs of nearby objects (see Appendix A.5)	$t_{\text{obj\_dist}} = 1200$	–

A.4 Hyperparameters of recent visual cropping methods

This subsection outlines the hyperparameters used for the baseline methods, i.e., for DC^2, SEAL, ViCrop, and ZoomEye.

For DC^2, the full evaluation code is not publicly available, and we were unable to reproduce their results using the provided demo code. Therefore, we report the performance metrics as stated in their paper and estimate the number of Forward Passes (FPs) based on the available demo. Specifically, we follow the procedure described in the paper: splitting the image into patches using the base resolution of the vision encoder (i.e., 336×336 for LLaVA-1.5) and merging patches via hierarchical clustering to improve efficiency. Although the FPs for DC^2 are estimated and may carry a high margin of error, the method remains less efficient than other baselines. This is further exacerbated by its region-wise captioning step, which makes it more computationally intensive than

other baselines, even when the number of FPs is comparable. Furthermore, DC^2 consistently underperforms compared to the other methods across all three datasets—V*Bench, HRBench-4K, and HRBench-8K—a trend also noted in ZoomEye's evaluation.

For SEAL, we use the hyperparameters described in their paper and the default configuration provided in their code. Specifically, we set the minimum search size to 224 and the minimum search scale to 4. For the visual search, we use a confidence lower bound of 0.3 and a confidence upper bound of 0.5. Regarding the target cue, we set the threshold to 6, the decay factor to 0.7, and the minimum threshold to 4. Note that these parameters were originally configured for the V*Bench dataset, and we did not adjust them for the other datasets.

For ViCrop, we select only the two best-performing variants from their work: `att-grad-high` and `rel-att-high`. Notably, both methods employ the high-resolution processing scheme `high`, which divides high-resolution images into a grid of $1K$ sub-images and computes importance maps for each sub-image individually. As a result, the computational overhead increases significantly with higher input resolutions.

For ZoomEye, we report more results per dataset–model combination than those presented in the original paper to offer a more comprehensive view of its efficiency–accuracy trade-offs. Specifically, we vary two key parameters: the number of sub-regions into which each region is split (2 or the default 4 crops), and the depth of the search tree (1, 2, and the default 5). All other hyperparameters are as specified in the ZoomEye paper.

A.5 Additional implementation details of FOCUS

We provide additional implementation details of FOCUS to ensure reproducibility, including how to construct the object relevance maps and how to perform the final VQA prediction.

Constructing object relevance maps We provide PyTorch-style pseudocode in Figure 6. For a detailed motivation and description of this method, see Section 3.

Final VQA prediction As explained in Section 3.3, we follow the inference strategy introduced in ZoomEye [27] when passing the selected ROIs to the MLLMs. In this appendix, we elaborate on the strategy used for global-view MLLMs, which typically accept only a single low-resolution image input.

For type-1 questions, we select the ROI with the highest confidence score for each target object. If the question involves multiple target objects, we compute the combined image region covering all targets and use this as the final VQA input. However, this approach may fail when the target objects are far apart, as the resulting combined region can become excessively large and include irrelevant background. To address this, a fallback strategy is taken. If the Euclidean distance between any pair of target objects exceeds a threshold $t_{\text{obj\_dist}}$ (e.g., 1200 pixels), we instead resize each individual ROI to a lower resolution and paste them onto an empty canvas based on their relative positions. This avoids including unnecessary background while maintaining spatial relationships. The resulting composite image is then used for the final VQA prediction. For type-2 questions, we first select all ROIs whose confidence scores exceed a predefined threshold. Overlapping regions are then merged. The merged ROIs are subsequently placed on a canvas using the pasting strategy described above.

In the case of global-local-view MLLMs, this workaround is unnecessary, as these models natively support multi-image reasoning without requiring canvas composition. Fine-grained details in local regions can be preserved by supplying them as separate image inputs. We leverage the model's text–image interleaved capabilities [2, 19, 20] by providing a global view - with all ROIs visually highlighted, alongside the individual high-confidence local crops.

```
# X: value features
# O: total number of LLM layers
# H: total number of attention heads/kv-cache heads
# N: sequence length
# D: hidden size of each token embedding
# s: total number of target tokens
# bmm: batch matrix multiplication
# a: grid size of vision encoder
# l: start of the selected layer
# L: end of the selected layer
# x_visual: indices of the visual tokens, [a**2,]
# x_target: indices of the target tokens, [s,]
# x_layer: indices of selected layers, [L-l,]

X = l2_normalize(X, dim=-1) # O, H, N, D

X = reshape(X) # O, N, H*D

# calculate cosine similarity
X = bmm(X, X.permute(0, 2, 1)) # O, N, N

# initialize relevance map
object_rel_map = ones_like(x_visual)

for target in x_target:

    # initialize pseudo-attn for each target token
    pseudo_attn = zeros_like(x_visual) # [a**2]

    for layer in x_layer:
        identity = eye(N) # [N, N]

        # add residual connection
        X_layer = (X[layer] + identity) / 2 # [N, N]

        # row-wise normalization
        X_layer /= X_layer.sum(dim=-1) # [N, N]

        layer_pseudo_attn = X_layer[target, x_visual] # [a**2]

        pseudo_attn += layer_pseudo_attn # [a**2]

    # global normalization
    pseudo_attn /= pseudo_attn.sum()

    # element-wise multiplication of pseudo-attentions
    object_rel_map *= pseudo_attn

object_rel_map = reshape(object_rel_map) # [a, a]
```

Figure 6: **PyTorch-style pseudocode for constructing object relevance maps in FOCUS.**

B Dataset statistics

This section provides a summary of the datasets used in our experiments. For each dataset, we report the average image resolution, the number of images, and the number of question–answer pairs in Table 6.

Table 6: **Statistics of the employed VQA datasets**

Attribute	V*Bench	HRBench-4K	HRBench-8K	MME-RealWorld-Lite
Avg. width	2,246	4,024	7,431	2,836
Avg. height	1,582	3,503	5,358	1,566
Images [#]	191	200	200	1,543
QA-pairs [#]	191	800	800	1,919

V*Bench [33] comprises images with an average resolution of 2K, specifically $2,256 \times 1,582$ pixels. The dataset includes two types of tasks: *direct attribute* and *spatial relation*. The *direct attribute* task involves identifying visual properties of a single object (e.g., color), making them single-target tasks. In contrast, the *spatial relation* task requires predicting the spatial relationship between two objects, thus constituting multiple-target tasks. In total, the dataset contains 191 images, each paired with a single question, resulting in 191 question-answer (QA) pairs.

HRBench [32] comprises two sub-datasets with average image resolutions of 4K and 8K, respectively. Both sub-datasets include two task types: *FSP* (Fine-grained Single-instance Perception) and *FCP* (Fine-grained Cross-instance Perception). *FSP* questions are single-target and focus on identifying fine-grained attributes of individual objects, whereas *FCP* questions are multiple-target and involve reasoning about spatial relationships between target objects. HRBench-8K contains full-resolution images with an average size of $7,431 \times 5,358$ pixels, while HRBench-4K provides cropped versions of these images, on average with $4,024 \times 3,503$ pixels. Each dataset contains one question per image. To improve evaluation robustness, HRBench permutes the positions of the answer options, yielding a total of 800 question–answer pairs across 200 images.

MME-RealWorld-Lite [37] is a real-world, fine-grained VQA dataset with an average image resolution of $2,836 \times 1,566$ pixels. It includes a diverse set of subtasks designed to evaluate the perception and reasoning capabilities of MLLMs across various domains, such as Autonomous Driving and OCR. The dataset is divided into two subsets: *Perception* and *Reasoning*. Each subset contains questions from multiple domains, including *OCR* (Optical Character Recognition), *RS* (Remote Sensing), *DT* (Diagram and Table), *MO* (Monitoring), and *AD* (Autonomous Driving), resulting in a total of nine tasks, as shown in Table 7. It comprises $1,543$ images, with some images associated with multiple questions, leading to a total of $1,919$ questions. The number of QA pairs per task is detailed in Table 7.

Table 7: **Number of question-answer (QA) pairs in MME-Realworld-Lite per domain**

	Perception					**Reasoning**			
	OCR	RS	DT	MO	AD	OCR	DT	MO	AD
QA-pairs [#]	250	150	100	319	350	100	100	150	400

C Further results

We include full numerical results in Appendix C.1, additional efficiency metrics in Appendix C.2, and a comparison of different inference schemes as well as performance discrepancies between reported and reproduced results in Appendix C.3.

C.1 Full results

We present the full results of ZoomEye [27], ViCrop [36], DC^2 [32], SEAL [33], and our method `FOCUS` on V*Bench, HRBench-4K, and HRBench-8K in Table 8. For MME-RealWorld-Lite [37], we compare ZoomEye, the vanilla baseline, and `FOCUS`, see Table 9.

As described in Section 4.1, we run experiments with $n_{\text{steps}} \in \{1, 2, 3, 4, 6, 8\}$. To better leverage the reasoning capabilities of MLLMs under higher computational budgets, we allow an `overrun` mode: if the logit l_{Yes} is lower than l_{No}—i.e., the MLLM responds `"No"` to the existence prompt for all of the top-n_{steps} ROIs—the model continues evaluating additional ROIs until it receives a `"Yes"` response. This mechanism improves the efficiency trade-off in many cases. For $n_{\text{steps}} \in \{1, 2\}$, we report results both with and without the `overrun` mechanism to provide a complete comparison in the low-computation setting.

In general, accuracy improves with increased computation budget for methods that support a configurable computation budget—such as ZoomEye and `FOCUS`. ZoomEye exhibits *exponential* scaling in the number of evaluated regions due to its hierarchical tree structure, leading to a significantly higher number of Forward Passes (FPs) when targeting high accuracy. In contrast, `FOCUS` constructs an object-aware relevance map and directly retrieves the most relevant regions, resulting in *linear* scaling with respect to the number of FPs.

C.2 Additional efficiency metrics

We report additional efficiency metrics—including average execution time and peak GPU memory usage per sample—on V*Bench. We compare SEAL, vanilla LLaVA-1.5-7B, vanilla LLaVA-OneVision-7B, and three training-free methods (ViCrop, ZoomEye, and `FOCUS`) using LLaVA-1.5 models.

As shown in Table 10, among the training-free methods, ZoomEye achieves the highest accuracy but suffers from poor efficiency due to its complex confidence prediction mechanism, which involves multiple question prompts and a hierarchical tree structure. This is reflected by its high number of Forward Passes (FPs) and long execution times. `FOCUS`, by contrast, leverages an object relevance map built from cached token similarities to directly identify relevant image regions. As a result, it requires only 25% of ZoomEye's FPs and execution time to reach comparable accuracy. ViCrop is slightly more efficient than `FOCUS` in terms of execution time and FPs, but it achieves lower accuracy and incurs the highest peak memory usage due to its incompatibility with efficient attention mechanisms. SEAL differs architecturally from LLaVA-1.5 and LLaVA-OneVision. Its dual-MLLM design makes it significantly slower and more memory-intensive than most methods in the comparison. In general, a lower number of FPs is associated with reduced execution time.

C.3 Inference scheme and performance discrepancy

We describe in the following the comparison between different inference schemes and the discrepancy between reported and reproduced performance.

Inference scheme: logits matching vs. open-ended generation Multiple-choice VQA requires selecting one of several fixed answer options. A common approach is open-ended generation [3, 13], where the prompt includes the question and options (e.g., `"(A) Red"`), and the MLLM generates the corresponding option letter. In contrast, SEAL [33] and ZoomEye [27] adopt an alternative scheme called logits matching on V*Bench [33]. In this method, the model is prompted multiple times, once for each answer option. Specifically, each answer option is reformulated as a sentence (e.g., `"(A) Red"` → `"The color of the car is red."`) which is then appended to the original question. The model is prompted with these reformulated question–option pairs and the image, and the answer option yielding the highest logit score for its target tokens is selected as the final prediction.

We noticed that ZoomEye utilizes logits matching on V*Bench but uses open-ended generation on the other three datasets, prompting us to investigate the impact of different inference schemes. We evaluate LLaVA-1.5 on V*Bench using both open-ended generation and logits matching across three methods: the vanilla baseline, ZoomEye, and `FOCUS` (ours). SEAL is excluded from this comparison, as it is not a training-free method. All other hyperparameters are kept constant.

As shown in Table 11, both the vanilla model and `FOCUS` achieve 0.5% and 1.6% higher accuracy, respectively, when using logits matching. This improvement likely stems from the fact that logits matching eliminates the need for strong instruction-following: models no longer need to explicitly generate the option letter, but

Table 8: **Full results of different models on fine-grained VQA benchmarks.** V*Bench comprises two tasks, namely direct attribute (Attr) and spatial relationship (Spatial). Similarly, HRBench consists of two tasks, i.e., Fine-grained Single-instance Perception (FSP) and Fine-grained Cross-instance Perception (FCP). The highest accuracy for each method-model combination is highlighted in bold. As DC^2 does not provide the complete evaluation code, we report the accuracy from the original paper and estimate the number of FPs following the procedure described in Appendix A.4.

Model	V*Bench Overall Acc. ↑ (Attr | Spatial) [%]	V*Bench FP [#] ↓	HRBench-4K Overall Acc. ↑ (FSP | FCP) [%]	HRBench-4K FP [#] ↓	HRBench-8K Overall Acc. ↑ (FSP | FCP) [%]	HRBench-8K FP [#] ↓
ZoomEye						
LLaVA-1.5-7B						
`Depth-1 (2 crops)`	50.26 (41.74 | 63.16)	12.50	36.25 (39.25 | 33.25)	11.54	33.88 (32.25 | 35.5)	11.55
`Depth-1 (4 crops)`	50.78 (41.74 | 64.47)	20.37	39.13 (44.75 | 33.5)	17.46	32.88 (32.75 | 33.00)	18.18
`Depth-2 (4 crops)`	71.20 (67.83 | 76.32)	44.54	47.75 (57.25 | 38.25)	35.60	42.13 (49.00 | 36.25)	39.15
`Depth-5 (4 crops)`	**77.48** (80.87 | 72.37)	48.63	**50.00** (63.25 | 36.75)	49.38	**49.00** (61.75 | 36.25)	59.64
LLaVA-OV-7B						
`Depth-1 (2 crops)`	75.92 (77.39 | 73.68)	10.52	61.38 (70.00 | 52.75)	9.46	59.00 (66.50 | 51.50)	9.02
`Depth-1 (4 crops)`	81.15 (81.74 | 80.26)	13.34	65.75 (80.75 | 50.75)	11.43	59.63 (69.00 | 50.25)	12.59
`Depth-2 (4 crops)`	90.05 (93.04 | 85.53)	21.08	66.13 (80.25 | 52.00)	20.42	65.63 (81.75 | 49.5)	22.31
`Depth-5 (4 crops)`	**91.10** (93.91 | 86.84)	23.98	**69.38** (84.5 | 54.25)	29.19	**69.00** (86.75 | 51.25)	36.75
ViCrop						
LLaVA-1.5-7B						
`rel-att-high`	**59.16** (58.26 | 60.53)	12.26	42.50 (51.50 | 33.50)	30.59	**39.38** (48.00 | 30.75)	78.60
`grad-att-high`	54.97 (53.04 | 57.90)	6.63	**44.25** (53.75 | 34.75)	15.80	38.38 (44.25 | 32.50)	39.80
DC^2						
LLaVA-v.1.5-7B	**57.00** (— | —)	18.18	**42.30** (— | —)	48.55	**39.50** (— | —)	77.02
SEAL	**73.68** (tbd | tbd)	25.53	**34.50** (39.00 | 30.00)	18.05	**33.50** (37.00 | 30.00)	16.96
`FOCUS` **(Ours)**						
LLaVA-1.5-7B						
`Steps-1 (no-overrun)`	51.30 (46.95 | 57.89)	1.47	41.13 (49.75 | 32.5)	3.14	40.63 (46.00 | 35.25)	3.14
`Steps-2 (no-overrun)`	57.07 (53.91 | 61.84)	4.25	46.5 (56.00 | 37.00)	5.41	40.63 (44.75 | 36.50)	5.41
`Steps-1 (overrun)`	64.40 (63.48 | 65.79)	4.86	42.25 (51.50 | 33.00)	4.99	42.38 (48.50 | 36.25)	5.08
`Steps-2 (overrun)`	66.49 (66.09 | 67.11)	5.70	45.88 (55.75 | 36.00)	5.87	43.25 (46.75 | 37.50)	6.04
`Steps-3 (overrun)`	67.01 (66.09 | 68.42)	6.79	48.63 (56.50 | 37.75)	6.76	44.13 (48.00 | 40.25)	9.07
`Steps-4 (overrun)`	68.06 (66.96 | 69.74)	8.27	49.25 (59.75 | 38.75)	10.14	**45.00** (50.25 | 39.75)	10.10
`Steps-6 (overrun)`	70.68 (70.43 | 71.05)	10.71	50.63 (62.25 | 39.00)	12.31	**45.00** (52.00 | 38.00)	12.23
`Steps-8 (overrun)`	**72.77** (72.17 | 73.68)	13.28	**51.75** (64.00 | 39.50)	14.49	44.13 (52.25 | 36.00)	14.41
LLaVA-OV-7B						
`Steps-1 (no-overrun)`	83.24 (87.82 | 76.31)	1.47	69.00 (82.25 | 55.75)	3.84	65.75 (77.00 | 54.50)	3.86
`Steps-2 (no-overrun)`	89.01 (92.17 | 84.21)	4.23	70.38 (84.00 | 56.75)	6.07	66.88 (78.00 | 55.75)	6.09
`Steps-1 (overrun)`	90.57 (93.04 | 86.84)	4.05	69.88 (85.75 | 54.00)	6.55	68.75 (81.00 | 56.50)	7.35
`Steps-2 (overrun)`	91.62 (93.93 | 88.15)	5.16	70.25 (86.50 | 54.00)	7.41	67.63 (79.00 | 56.25)	8.06
`Steps-3 (overrun)`	91.62 (93.91 | 88.15)	6.37	70.00 (85.75 | 54.25)	8.34	66.88 (78.00 | 55.75)	8.93
`Steps-4 (overrun)`	**92.15** (93.91 | 89.47)	7.63	70.75 (85.75 | 55.75)	9.32	68.38 (80.75 | 56.00)	9.86
`Steps-6 (overrun)`	**92.15** (93.91 | 89.47)	10.22	70.63 (85.75 | 55.50)	11.33	68.88 (82.50 | 55.25)	11.81
`Steps-8 (overrun)`	91.62 (93.04 | 89.47)	12.84	**71.13** (86.75 | 55.50)	13.41	**69.63** (83.50 | 55.75)	13.83

Table 9: **Detailed results on the MME-RealWorld-Lite dataset.** Accuracy is reported both per subtask and on average per subset. The highest accuracy for each subtask is highlighted in bold.

		Task	*LLaVA-OV-7B* Vanilla Acc. [%]	ZoomEye Acc. [%]	FOCUS (Ours) Acc. [%]
Perception	AD	$\texttt{Motion}_{\texttt{multi-pedestrians}}$	22.00	28.00	**34.00**
		$\texttt{Motion}_{\texttt{multi-vehicles}}$	**46.00**	38.00	40.00
		$\texttt{Motion}_{\texttt{pedestrian}}$	24.00	**46.00**	44.00
		$\texttt{Motion}_{\texttt{vehicle}}$	24.00	**54.00**	34.00
		$\texttt{Object}_{\texttt{count}}$	36.00	**42.00**	40.00
		$\texttt{Object}_{\texttt{identify}}$	**78.00**	74.00	70.00
		$\texttt{Visual}_{\texttt{traffic-signal}}$	62.00	**78.00**	68.00
	DT	Diagram	70.00	**80.00**	60.00
		Table	60.00	**68.00**	56.00
	MO	$\texttt{Person}_{\texttt{color}}$	32.00	40.00	**56.00**
		$\texttt{Person}_{\texttt{counting}}$	32.00	**38.00**	**38.00**
		$\texttt{Person}_{\texttt{orientation}}$	10.53	**15.79**	10.53
		$\texttt{Vehicle}_{\texttt{color}}$	46.00	**60.00**	52.00
		$\texttt{Vehicle}_{\texttt{counting}}$	56.00	56.00	**58.00**
		$\texttt{Vehicle}_{\texttt{location}}$	**38.00**	28.00	28.00
		$\texttt{Vehicle}_{\texttt{orientation}}$	12.00	20.00	**26.00**
	OCR	Advert & product	82.00	82.00	**88.00**
		Book map poster	**78.00**	70.00	74.00
		License	**88.00**	86.00	**88.00**
		Phone & address	82.00	**96.00**	90.00
		Text recognition	78.00	72.00	**80.00**
	RS	Color	60.00	**64.00**	**64.00**
		Count	34.00	**40.00**	20.00
		Position	**62.00**	50.00	56.00
		Average	52.01	**56.29**	54.15
Reasoning	AD	$\texttt{Attention}_{\texttt{traffic-signal}}$	**74.00**	72.00	**74.00**
		$\texttt{Intention}_{\texttt{ego}}$	**26.00**	24.00	22.00
		$\texttt{Intention}_{\texttt{pedestrian}}$	48.00	50.00	**54.00**
		$\texttt{Intention}_{\texttt{vehicle}}$	26.00	**42.00**	40.00
		$\texttt{Interaction}_{\texttt{ego-2-pedestrian}}$	20.00	**28.00**	22.00
		$\texttt{Interaction}_{\texttt{ego-2-traffic-signal}}$	28.00	**30.00**	22.00
		$\texttt{Interaction}_{\texttt{ego-2-vehicle}}$	26.00	22.00	**26.00**
		$\texttt{Interaction}_{\texttt{other-2-other}}$	10.00	**12.00**	8.00
	DT	Diagram	40.00	54.00	**58.00**
		Table	40.00	44.00	**46.00**
	MO	Calculate	42.00	46.00	**50.00**
		Intention	26.00	26.00	**42.00**
		Property	64.00	**66.00**	62.00
	OCR	Character identification	72.00	64.00	**74.00**
		Scene understanding	**72.00**	64.00	68.00
		Average	40.93	43.20	**44.53**

Table 10: **Additional performance and efficiency metrics on V*Bench.** In the last three rows, the best-performing method is highlighted in bold and the runner-up is underlined.

Model	**Acc.**$[\%]\uparrow$	**FP** $[\#]\downarrow$	**Avg. execution time** $[s]\downarrow$	**Avg. peak memory** $[GB]\downarrow$
SEAL	73.68	25.53	9.16	27.34
LLaVA-OV-7B	74.46	-	1.30	19.64
LLaVA-1.5-7B	47.64	-	0.25	13.57
+ w/ ViCrop	59.16	**12.26**	**1.36**	19.93
+ w/ ZoomEye	**77.49**	48.63	11.26	**14.24**
+ w/ Ours	<u>72.77</u>	<u>13.28</u>	<u>2.19</u>	<u>14.91</u>

Table 11: **Comparison of different inference schemes across models.** Each result includes accuracy, execution time, and number of forward passes for the visual search (FP). The highest accuracy in a column is highlighted in bold.

Inference Scheme	Vanilla	ZoomEye			FOCUS (Ours)		
	Acc. [%]↑	Acc. [%]↑	Exec. time[s]↓	FP [#] ↓	Acc. [%]↑	Exec. time[s]↓	FP [#] ↓
Logits matching	**48.16**	**77.49**	11.26	48.63	**74.35**	2.76	13.28
Open-ended generation	47.64	72.25	9.26	36.94	72.77	2.19	13.28

Table 12: **Reported vs. reproduced accuracy across fine-grained VQA datasets.** A dash indicates that no evaluation on that benchmark was performed in the original work.

Model	V*Bench		HRBench-4K		HRBench-8K	
	Reported Acc. [%]	Reprod. Acc. [%]	Reported Acc. [%]	Reprod. Acc. [%]	Reported Acc. [%]	Reprod. Acc. [%]
ZoomEye						
LLaVA-1.5-7B	83.25	77.48	53.25	50.00	50.75	49.00
LLaVA-OV-7B	90.58	91.10	69.63	69.38	69.25	69.00
ViCrop						
LLaVA-1.5-7B						
`rel-att-high`	62.30	59.16	–	42.50	–	39.38
`grad-att-high`	57.07	54.97	–	44.25	–	38.38
SEAL	75.39	73.68	–	34.50	–	33.50

instead compare the semantic content of full answer statements. This makes the inference process more robust, particularly for models with weaker generative alignment.

In contrast, ZoomEye's accuracy drops by over 5% and its execution time decreases by approximately 2 seconds when switching to open-ended generation. Given that FOCUS shows a 1.6% accuracy difference and a 0.57-second reduction in execution time under the same scheme change, we attribute that portion of ZoomEye's decline to the scheme itself. The remaining gap—both in accuracy and runtime—can likely be attributed to suboptimal tuning in the open-ended setting, as all hyperparameters were held constant. The changed inference mode likely alters the model confidence, which may lead to premature termination of tree search (indicated by the lower number of FPs) and increased prediction instability.

Despite the potential accuracy gains of logits matching, we adopt open-ended generation for FOCUS across all datasets. This is done for two reasons. First, the effectiveness of logits matching depends heavily on the quality of option reformulations, which are not always available or consistent across datasets. This limits its generalizability and makes cross-dataset comparisons less reliable. Second, logits matching requires one Forward Pass (FP) per answer option, compared to only a single FP in open-ended generation. Although some acceleration of the logits matching scheme is implemented by caching question tokens, it still increases inference time (see Table 11). Therefore, we report FOCUS results using open-ended generation in the main paper. For SEAL and ZoomEye, we preserve their original inference schemes. Their respectively reported V*Bench performance is based on logits matching.

Discrepancy between reported and reproduced performance For the baseline methods, i.e., SEAL [33], ViCrop [36], and ZoomEye [27], we use the official implementations provided in their respective repositories. We strictly follow the original configurations, including software environments and data structures, as specified by each work. As shown in Table 12, we observe some discrepancies between the reported and reproduced performance, most notably for LLaVA-1.5 with ZoomEye. We hypothesized this may be linked to the use of different efficient attention backends (e.g., FlashAttention-2 [8] vs. PyTorch's SDPA[2]). However, the observed deviation with different attention implementations is small, less than 1% on V*Bench—thus, further root causes of this deviation seem to exist, that however remain unclear. To ensure fair and consistent comparisons under a unified evaluation setup, we always report the reproduced results for these benchmark methods in the main paper.

[2]See `https://docs.pytorch.org/tutorials/intermediate/scaled_dot_product_attention_tutorial.html`

D Further qualitative examples

This section provides additional qualitative examples that highlight both the strengths and limitations of FOCUS when applied to LLaVA-1.5 (see Figure 7) and LLaVA-OneVision (see Figure 8). For improved clarity in visualization, we use a reduced $k = 10$, differing from the values specified in Appendix A.3.

Figure 7 **(I)** showcases a type-1 single-target task with FOCUS and LLaVA-1.5. By leveraging the MLLM's internal representations, FOCUS identifies a relevant crop that highlights the color of small candles, correcting the model's initial VQA response. The ROI ranking mechanism demonstrates robustness to noise in the object relevance map by assigning the highest confidence to the originally fourth-ranked region. Moving to a type-1 multiple-target task, Figure 7 **(II)** illustrates how FOCUS identifies both the `person in the red jacket` and the `large tree` using in-context learning. The object relevance maps clearly localize both targets. Since LLaVA-1.5 accepts only single-image inputs, the selected ROIs are stitched together—a strategy detailed in Appendix A.5—to avoid excessive image size or irrelevant content. Despite this, the model fails to answer correctly, likely due to limited spatial reasoning. A type-2 counting task is depicted in Figure 7 **(III)**. Here, FOCUS successfully locates both chairs in the image. As in the previous example, the regions are combined to form a single input for LLaVA-1.5. This enables the model to correctly answer the VQA query, which it could not do using the global view. Figure 7 **(IV)**, finally, presents another failure case with LLaVA-1.5, underscoring the limitation discussed in Section 4.5. The example, drawn from the HRBench-8K dataset (see Table 6), involves a high-resolution image where the sign is too small to be detected via the low-resolution object relevance map. Consequently, FOCUS selects an incorrect region and fails to improve the VQA result.

Turning to LLaVA-OneVision, Figure 8 **(I)** features a type-1 single-target task. While vanilla LLaVA-OneVision fails to answer the question about the speed limit sign, FOCUS successfully identifies the relevant region using internal representations. By isolating this region (see selected ROI), the model is able to generate the correct VQA response. Figure 8 **(II)** explores a type-1 multiple-target task. Despite the relatively large size of the relevant regions, vanilla LLaVA-OneVision does not answer correctly. FOCUS identifies the appropriate areas, generates a combined image region, and creates one local crop per relevant object. The relevant regions are highlighted in the image of the combined ROIs with rectangles, helping reduce background noise and enabling the model to answer the VQA task correctly. Next, a type-2 counting task is shown in Figure 7 **(III)** with LLaVA-OneVision. Vanilla LLaVA-OneVision fails to count the number of computers accurately. FOCUS identifies five computers in total, assisting the model in producing the correct answer. However, it only detects four correctly—missing one and mistakenly counting another one twice. Finally, Figure 7 **(IV)** illustrates a failure case with LLaVA-OneVision. Despite access to a high-resolution object relevance map, FOCUS fails to detect the region associated with the umbrella. As a result, it does not provide the necessary input for the MLLM to answer the VQA example correctly.

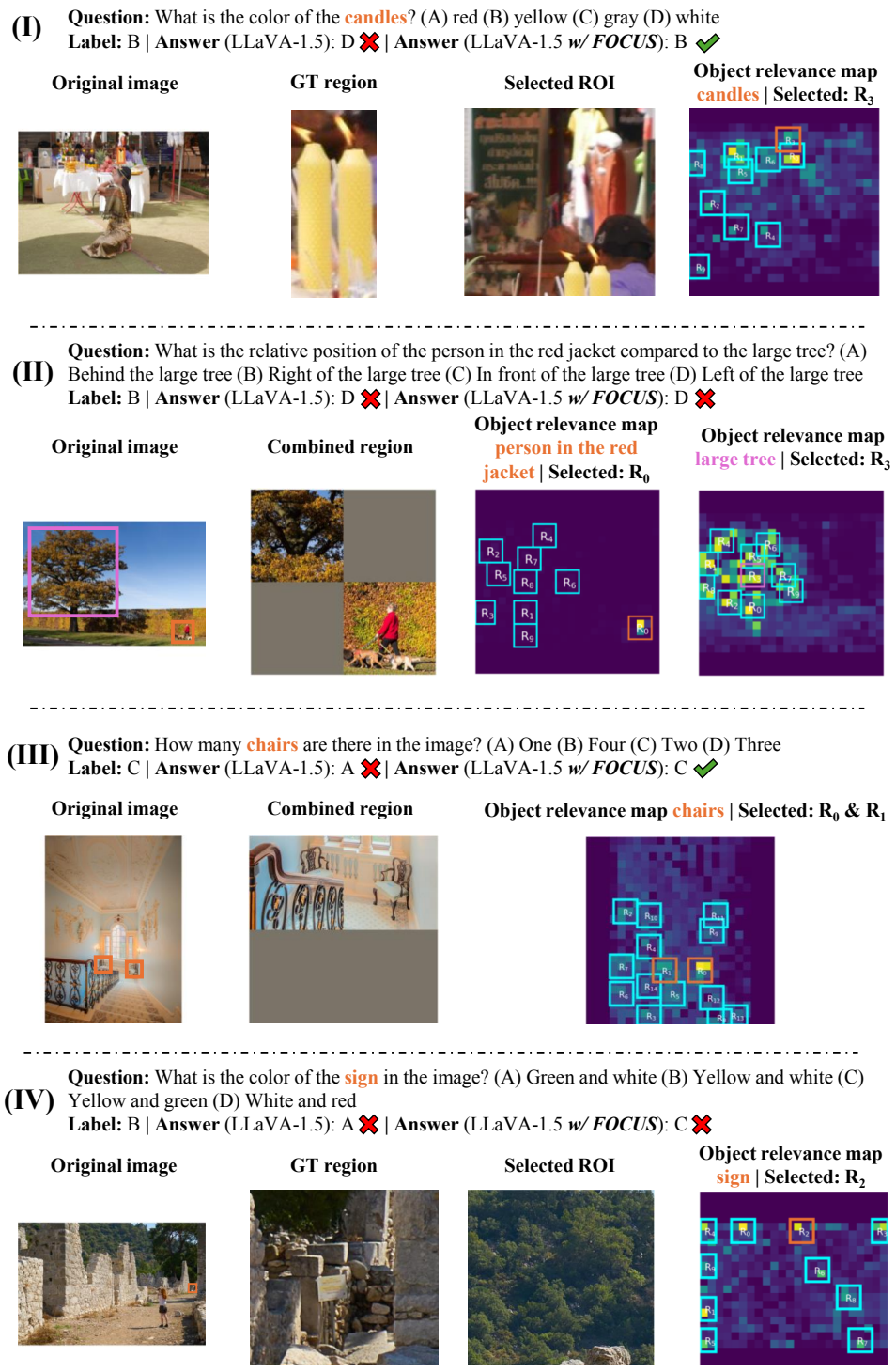

Figure 7: **Further qualitative examples of FOCUS with LLaVA-1.5.** We provide examples for single-object **(I)**, multi-object **(II)**, a type-2 question **(III)**, and a failure case **(IV)**. Note that we do not adjust the aspect ratio of the images for LLaVA-1.5. Therefore, there are some padding areas in the object relevance maps. Additionally, we manually highlight the relevant regions in the original image to facilitate easier localization of the ground truth area for the reader and these annotations are not included in the input for the MLLM.

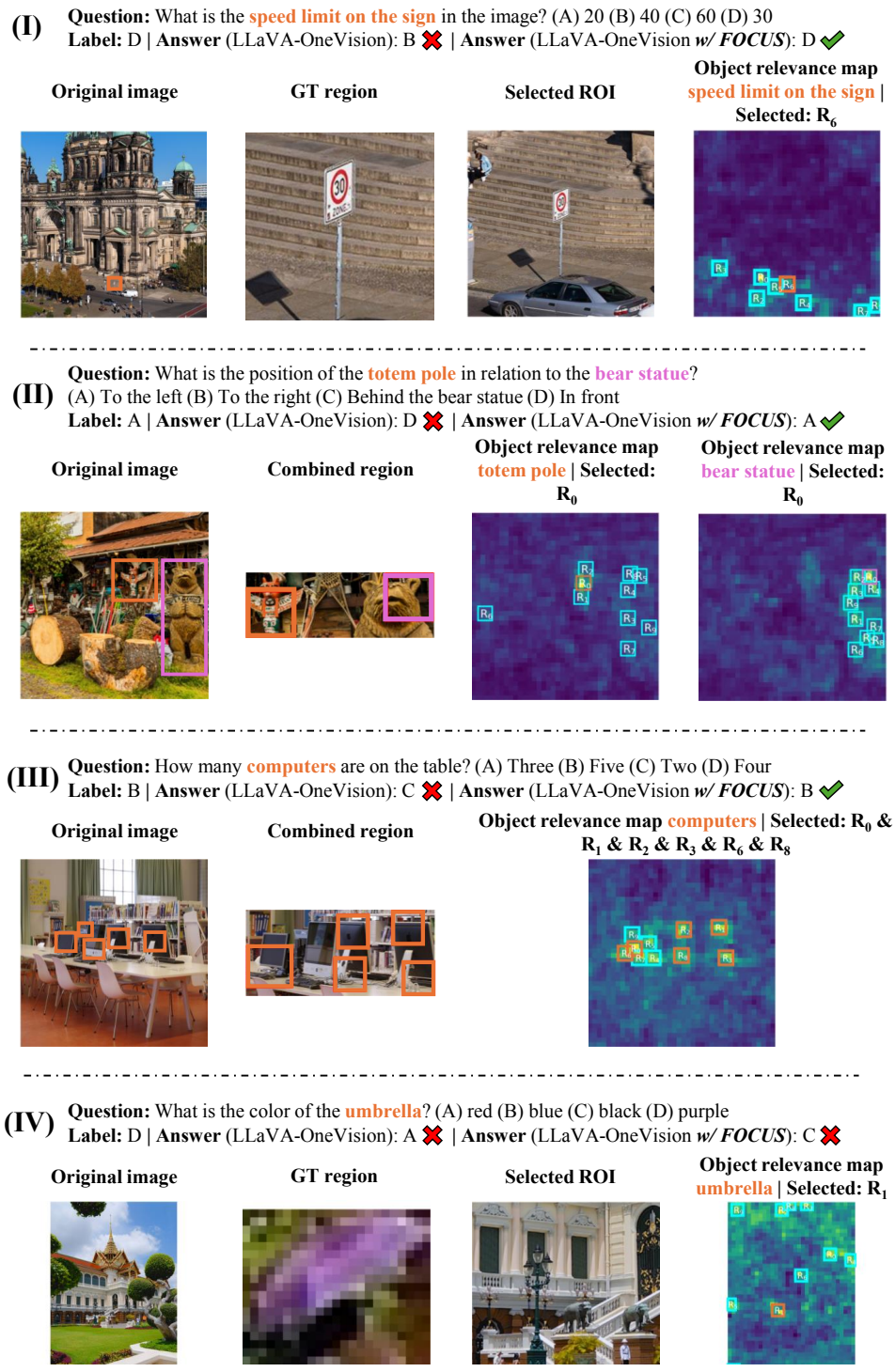

Figure 8: **Further qualitative examples of FOCUS with LLaVA-OneVision.** We provide examples for single-object **(I)**, multi-object **(II)**, a type-2 question **(III)**, and a failure case **(IV)**. Note that we manually highlight the relevant regions in the original image to facilitate easier localization of the ground truth area for the reader and these annotations are not included in the input for the MLLM.