FocusUI: Efficient UI Grounding via Position-Preserving Visual Token Selection

Supplementary Material

A. Implementation Details

A.1. Training Data

Raw Dataset Our training set compiles several public high-quality GUI datasets, following GUI-Actor. To ensure fair evaluation, samples from Wave-UI that overlap with the test sets of downstream tasks are excluded.

Refining Annotation Quality We apply OmniParser V2 [7] to filter samples whose IoU between ground-truth and OmniParser detected boxes is below 0.3. This results in a reduction of 22.9% in the number of elements. The final training statistics are in Tab. 1.

Dataset	#Screenshots	#Elements	Platform
UGround [4]	775K	8M	Web
GUI-Env [10]	70K	262K	Web
GUI-Act [10]	13K	42K	Web
AndroidControl [6]	47K	47K	Android
AMEX [1]	100K	1.2M	Android
Wave-UI	7K	50K	Hybrid
Total (Raw Dataset)	1012K	9.6M	–
Total (After Filtering)	**976K**	**7.4M**	–

Table 1. Statistics of training datasets used for FocusUI.

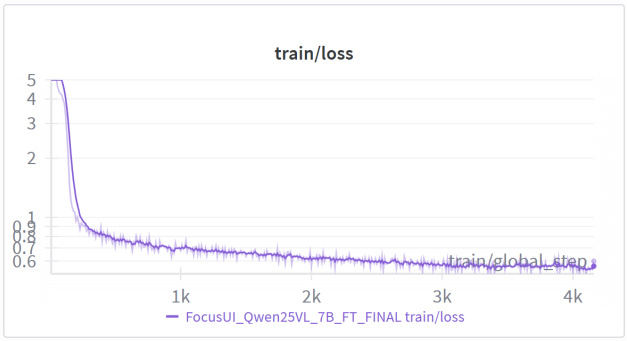
A.2. Training Recipe

We train FocusUI on 8× NVIDIA H200 GPUs using bfloat16 precision, DeepSpeed ZeRO-2 [9], and FlashAttention-2 [3]. The effective batch size per GPU is set to 32 (with gradient accumulation of 4), and `max_pixels` is set to 5720064, matching GUI-Actor. Training proceeds in two stages:

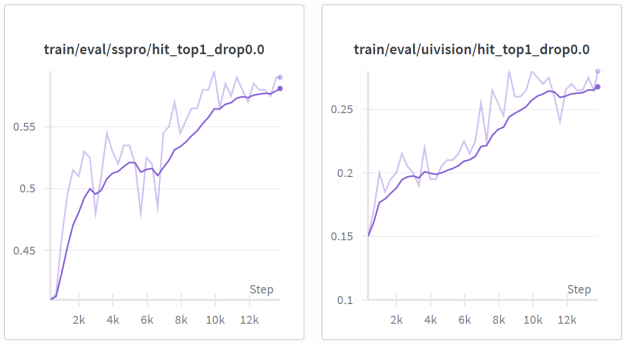
Stage 1: Saliency Scorer Pre-training. We pretrain the randomly initialized Query-Guided Saliency Scorer for 1 epoch with a learning rate of $1e-4$. This takes ∼12 hours for both 3B and 7B models.

Stage 2: Full Model Fine-tuning. We fine-tune all parameters for 1 epoch with a learning rate of $5e-6$. This takes ∼36 hours for the 3B model and ∼48 hours for the 7B model.

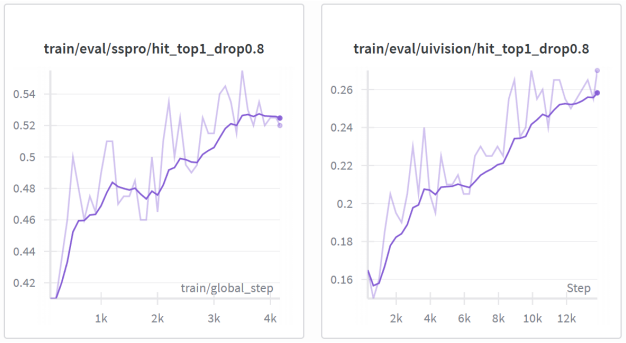
For reproducibility, we use the final checkpoint for all models. We also provide the full Weights & Biases (WandB) logs for all trained models; the loss and evaluation curves during the training of FocusUI-7B are shown in Fig. 1.



(a) **Total Loss** curve during training.



(b) **Evaluation:** ScreenSpot-Pro and UI-Vision with **retain ratio = 100%**.



(c) **Evaluation:** ScreenSpot-Pro and UI-Vision with **retain ratio = 20%**.

Figure 1. WandB loss and evaluation results of FocusUI-7B.

A.3. UI Grounding Benchmarks

We evaluate on four public benchmarks containing screenshots paired with instructions: **ScreenSpot-V2** [11], **ScreenSpot-Pro** [5], **OSWorld-G** [13] and **UI-Vision** [8]. The statistics of these benchmarks are shown in Tab. 2.

ScreenSpot-V2 [11]. A refined version of ScreenSpot [2] with 1,272 samples across mobile, desktop, and web environments.

ScreenSpot-Pro [5]. Contains 1,581 samples from 23 professional applications, targeting high-resolution interfaces

and complex layouts to test generalization.

OSWorld-G [13]. Sampled from OSWorld [12], this benchmark includes 564 samples categorized by task type (text matching, element recognition, layout understanding, fine-grained manipulation, and refusal).

UI-Vision [8]. A desktop-centric benchmark with 5,790 samples from 83 applications, evaluating element grounding, layout grounding, and action prediction.

Benchmark	#Samples	Avg Res.	Max Res.	Platform
ScreenSpot-V2	1272	1725×1657	2880×1800	Hybrid
ScreenSpot-Pro	1581	3267×1727	6016×3384	Desktop
OSWorld-G	564	1696×955	1920×1080	Desktop
UI-Vision	5790	1851×1034	3360×2036	Desktop

Table 2. Overview of testing benchmarks used in this work.

B. Discussions

B.1. Visual Redundancy Analysis

Tab. 3 details the token share statistics from **Study 1** (in Fig. 1(b)). Using default model settings on the ScreenSpot-Pro evaluation, we find that visual tokens occupy at least 84.3% of the sequence across the studied benchmarks, confirming significant visual redundancy in UI grounding tasks.

Benchmark	Model	#Sys. Tokens	#Vis. Tokens	#Inst. Tokens	Vis. Token Share
ScreenSpot-V2	Jedi-1080p	397	2348	4.5	85.4%
	GUI-Actor	90	3506	4.5	97.1%
ScreenSpot-Pro	Jedi-1080p	397	2629	5.2	86.7%
	GUI-Actor	90	5801	5.2	98.1%
OSWorld-G	Jedi-1080p	397	2244	21.3	84.3%
	GUI-Actor	90	2244	21.3	95.3%
UI-Vision	Jedi-1080p	397	2249	9.9	84.7%
	GUI-Actor	90	2566	9.9	96.3%

Table 3. Token share statistics for **Study 1** shown in Fig. 1(b).

B.2. Position Sensitivity Analysis

Tab. 4 shows the detailed results of **Study 2** (in Fig. 1(c)), comparing FocusUI with UI grounding models integrated with advanced visual token pruning methods.

C. More Experimental Results

C.1. Effective Visual Selection: Patch Recall@K%

We verify the effectiveness of our visual token selection using **Patch Recall@K%** ↑, defined as the fraction of ground-truth regions captured within the top K% of saliency-ranked patches (*i.e.*, the top-K visual tokens):

$$\text{Patch Recall@}K\% = \frac{|\,\text{GT positive regions in top } K\%\,|}{|\,\text{Total GT positive regions}\,|}$$

%Ret. Ratio	Model + Pruning Method	SS-V2 Avg	SS-Pro Avg	OSW-G Avg
100%	Qwen2.5-VL-3B	81.5	26.1	27.3
50%	+ Fast-V	43.5	13.9	14.3
	+ HiPrune	80.4	20.3	26.2
	+ Vision-Zip	81.0	21.0	27.1
30%	+ Fast-V	38.6	4.8	14.4
	+ HiPrune	72.0	18.0	20.4
	+ Vision-Zip	75.4	18.9	23.0
100%	Jedi-3B	88.9	36.1	48.8
50%	+ Fast-V	50.3	20.4	25.3
	+ HiPrune	88.3	32.8	46.4
	+ Vision-Zip	88.1	32.9	46.6
30%	+ Fast-V	51.0	14.1	23.9
	+ HiPrune	80.9	26.2	40.4
	+ Vision-Zip	82.8	28.8	41.5
100%	FocusUI-3B	91.5	43.8	53.4
50%	+ Full Settings	91.4	42.3	54.6
30%	+ Full Settings	91.0	40.6	51.8

Table 4. Detailed comparison with general visual token pruning methods for **Study 2** shown in Fig. 1(c).

where ground-truth positive regions correspond to the area of UI elements paired with the given instruction. We evaluate $K \in \{5\%, 10\%, 25\%, 50\%\}$. Additionally, we report the **Full Coverage Budget** ↓, the percentage of visual tokens needed to fully cover the ground-truth elements. Results are shown in Table 5.

Model	Patch Recall@K% ↑ @5%	@10%	@25%	@50%	Avg ↑	Full Coverage Budget ↓
Zero-shot Baselines						
Random	0.05	0.11	0.26	0.51	0.23	0.85
CLIP	0.12	0.21	0.41	0.65	0.35	0.61
our Query-Guided Saliency Scorer						
FocusUI-3B	0.39	0.56	0.83	0.96	0.69	0.25
FocusUI-7B	0.43	0.60	0.84	0.97	0.71	0.24

Table 5. **Patch Recall@K%** ↑ and **Full Coverage Budget** ↓ performance comparison on the ScreenSpot-Pro benchmark.

C.2. Analysis and Ablation of PosPad

We further analyze PosPad placement within contiguous sequences of dropped visual tokens, comparing three variants: (i) *sequence-first*, (ii) *sequence-middle*, and (iii) *sequence-end* (our proposed PosPad).

Tab. 6 reports the ablation study varying the visual token retention ratio r and where the PosPad token is placed. Across all retention ratios, placing PosPad at the end of each dropped sequence achieves the best performance, especially under low retention ratios. This suggests that *where*

the POSPAD token is placed in the original sequence matters for preserving the original spatial layout.

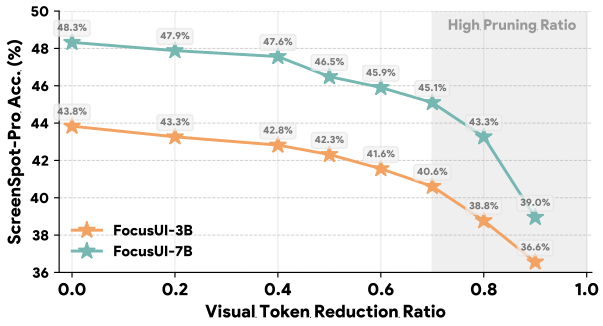
The empirical ablation results confirm our intuition: placing POSPAD at the sequence end is most compatible with the raster-scan ordering used by the vision encoder and M-RoPE. In contrast, placing the marker at the beginning or middle of the sequence pulls the whole region toward earlier positions, making it harder for the LM decoder to align with the original spatial structure.

Sequence Type	SS-Pro Avg.			
	$r = 100\%$	$r = 75\%$	$r = 50\%$	$r = 25\%$
sequence-first	42.0	41.8	39.8	36.8
sequence-middle	41.2	41.1	38.7	33.5
sequence-end (POSPAD)	42.3	42.1	40.4	37.7

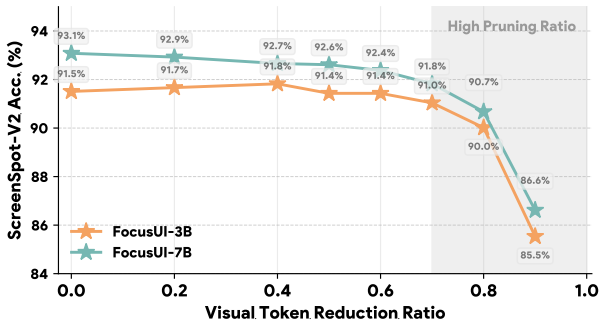
Table 6. Ablation of the placement of POSPAD tokens.

C.3. Detailed Performance vs. Retention Ratio

Fig. 2 presents the detailed performance of FOCUSUI on ScreenSpot-V2 and ScreenSpot-Pro across varying retention ratios. The results demonstrate that FOCUSUI maintains high UI grounding accuracy even with significant visual token reduction.



(a) Performance vs. retention ratio on ScreenSpot-Pro.



(b) Performance vs. retention ratio on ScreenSpot-V2.

Figure 2. Evaluation results under different retention ratios.

C.4. Qualitative Examples

Fig. 3 visualizes saliency maps on ScreenSpot-V2 and ScreenSpot-Pro, showing that our Query-Guided Saliency Scorer effectively highlights instruction-relevant regions while suppressing the background. We find that for straightforward tasks (Fig. 3 (a, d)), saliency scores peak significantly at the ground-truth locations. In more complex scenarios (Fig. 3 (b, c)), while the scores may be less concentrated, the model still successfully distinguishes potential targets from irrelevant background elements.

D. Prompt Templates

FOCUSUI (with Qwen2.5-VL base model) Below is the system prompt for FOCUSUI-3B and FOCUSUI-7B.

```
You are a GUI agent. Given a screenshot of the
current GUI and a human instruction, your task is
 to locate the screen element that corresponds to
 the instruction. You should output a PyAutoGUI
action that performs a click on the correct
position. To indicate the click location, we will
 use some special tokens, which is used to refer
to a visual patch later. For example, you can
output: pyautogui.click(<your_special_token_here
>).
```

FOCUSUI (with Qwen3-VL base model) Below is the system prompt for FOCUSUI-QWEN3-VL-2B.

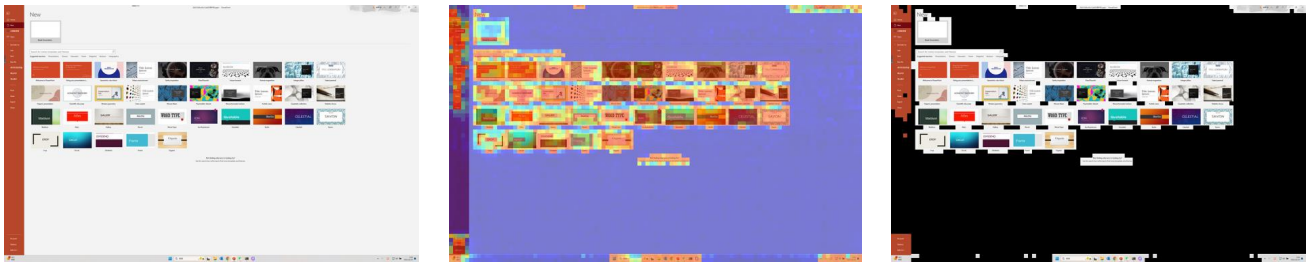
```
You are a GUI agent. Your task is to locate the
screen element that corresponds to the
instruction. You should not call any external
tools. Output only the coordinate of one point in
 your response. Format: (x, y)
```

Qwen2.5-VL Below is the system prompt for evaluating Qwen2.5-VL models.

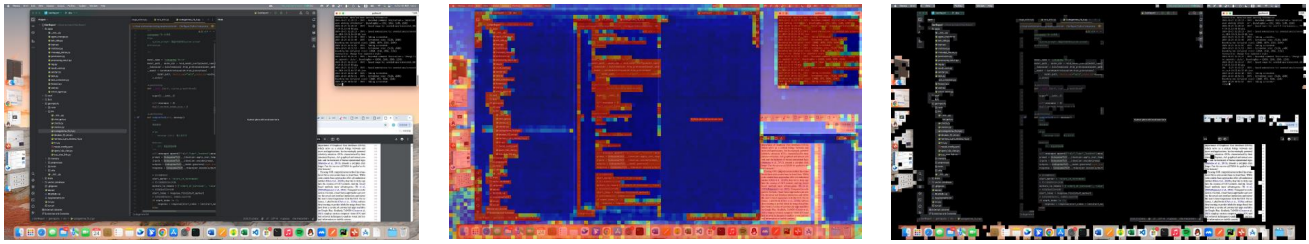
```
You are a GUI agent. Your task is to locate the
screen element that corresponds to the
instruction. You should not call any external
tools. Output only the coordinate of one point in
 your response. Format: (x, y)
```

Jedi and Qwen3-VL Below is the system prompt for evaluating Jedi-3B, Jedi-7B and Qwen3-VL models.

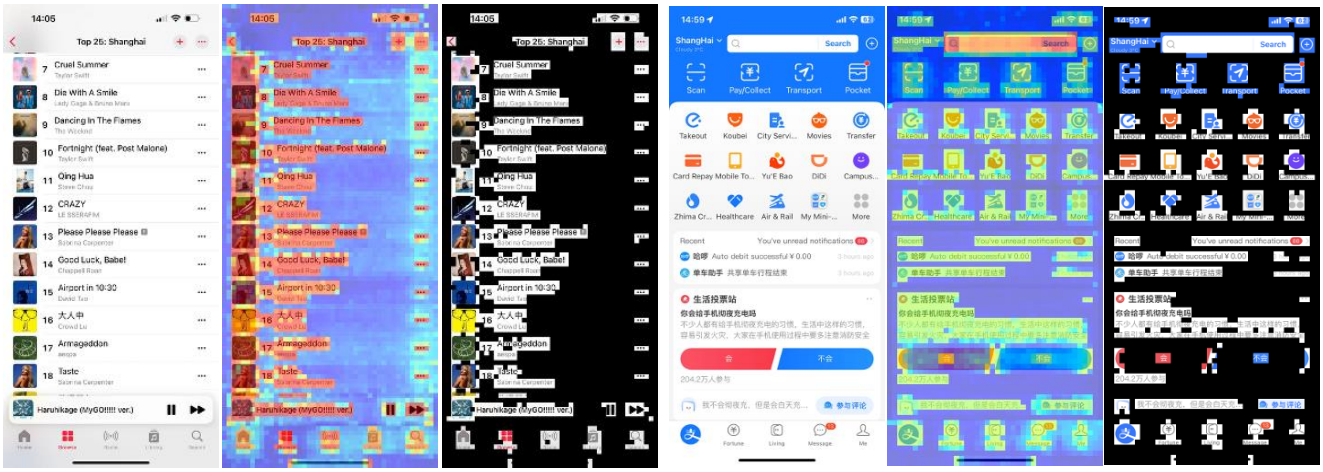
```
# Tools
You may call one or more functions to assist with
 the user query.

You are provided with function signatures within
<tools></tools> XML tags:
<tools>
{"type": "function", "function": {"name": "
computer_use", "description": "Use a mouse to
interact with a computer.\n* The screen's
resolution is {screen_width}x{screen_height}.\n*
Make sure to click any buttons, links, icons, etc
```



(a) Instruction query: "Create a Psychedelic vibrant presentation."



(b) Instruction query: "View hierarchy."



(c) Instruction query: "Pause the playing music."

(d) Instruction query: "Click the search bar."

Figure 3. Qualitative examples. **Left:** original screenshot; **Middle:** predicted saliency map; and **Right:** visual token selection results.

```
 with the cursor tip in the center of the element
. Don't click boxes on their edges unless asked.\
n* you can only use the left_click and mouse_move
 action to interact with the computer. if you can
't find the element, you should terminate the
task and report the failure.", "parameters": {"
properties": {"action": {"description": "The
action to perform. The available actions are:\n*
`mouse_move`: Move the cursor to a specified (x,
y) pixel coordinate on the screen.\n* `left_click
`: Click the left mouse button with coordinate (x
, y).\n* `terminate`: Terminate the current task
and report its completion status.", "enum": ["
mouse_move", "left_click"], "type": "string"}, "
coordinate": {"description": "(x, y): The x (
pixels from the left edge) and y (pixels from the
 top edge) coordinates to move the mouse to.
Required only by `action=mouse_move` and `action=
left_click`.", "type": "array"}, "status": {"
description": "The status of the task. Required
only by `action=terminate`.", "type": "string", "
enum": ["success", "failure"]}}, "required": ["
action"], "type": "object"}}}
</tools>

For each function call, return a json object with
 function name and arguments within <tool_call></
tool_call> XML tags:
<tool_call>
{"name": <function-name>, "arguments": <args-json
-object>}
</tool_call>
```

References

[1] Yuxiang Chai, Siyuan Huang, Yazhe Niu, Han Xiao, Liang Liu, Dingyu Zhang, Peng Gao, Shuai Ren, and Hongsheng Li. Amex: Android multi-annotation expo dataset for mobile gui agents. *arXiv preprint arXiv:2407.17490*, 2024. 1

[2] Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. Seeclick: Harnessing gui grounding for advanced visual GUI agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9313–9332, 2024. 1

[3] Tri Dao, Daniel Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems*, 2022. 1

[4] Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the digital world as humans do: Universal visual grounding for GUI agents. In *The Thirteenth International Conference on Learning Representations*, 2025. 1

[5] Kaixin Li, Ziyang Meng, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng Chua. Screenspot-pro: GUI grounding for professional high-resolution computer use. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 8778–8786, 2025. 1

[6] Wei Li, William E Bishop, Alice Li, Christopher Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the effects of data scale on ui control agents. *Advances in Neural Information Processing Systems*, 37: 92130–92154, 2024. 1

[7] Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. Omniparser for pure vision based GUI agent. *arXiv preprint arXiv:2408.00203*, 2024. 1

[8] Shravan Nayak, Xiangru Jian, Kevin Qinghong Lin, Juan A Rodriguez, Montek Kalsi, Nicolas Chapados, M Tamer Özsu, Aishwarya Agrawal, David Vazquez, Christopher Pal, et al. UI-Vision: A desktop-centric GUI benchmark for visual perception and interaction. In *Forty-second International Conference on Machine Learning*, 2025. 1, 2

[9] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020. 1

[10] Qianhui Wu, Kanzhi Cheng, Rui Yang, Chaoyun Zhang, Jianwei Yang, Huiqiang Jiang, Jian Mu, Baolin Peng, Bo Qiao, Reuben Tan, et al. GUI-Actor: Coordinate-free visual grounding for GUI agents. *arXiv preprint arXiv:2506.03143*, 2025. 1

[11] Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, et al. OS-ATLAS: Foundation action model for generalist GUI agents. In *The Thirteenth International Conference on Learning Representations*, 2024. 1

[12] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024. 2

[13] Tianbao Xie, Jiaqi Deng, Xiaochuan Li, Junlin Yang, Haoyuan Wu, Jixuan Chen, Wenjing Hu, Xinyuan Wang, Yuhui Xu, Zekun Wang, Yiheng Xu, Junli Wang, Doyen Sahoo, Tao Yu, and Caiming Xiong. Scaling computer-use grounding via user interface decomposition and synthesis. *Advances in Neural Information Processing Systems*, 2025. 1, 2