GUI-Bee: Align GUI Action Grounding to Novel Environments via Autonomous Exploration

Yue Fan [1], Handong Zhao [3], Ruiyi Zhang [3], Yu Shen [3], Xin Eric Wang [*1,2], Gang Wu [*3]
[1] UC Santa Cruz [2] UC Santa Barbara [3] Adobe Research
yfan71@ucsc.edu; ericxwang@ucsb.edu; {hazhao, ruizhang, shenyu, gawu}@adobe.com

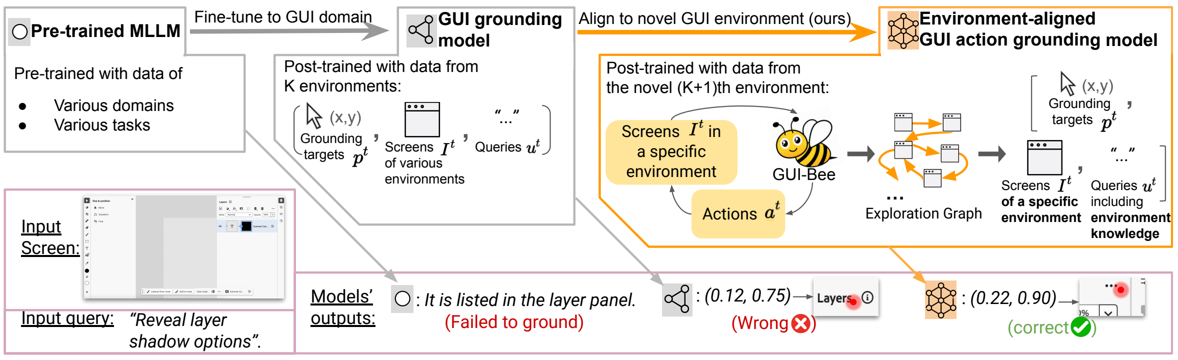

Pre-trained MLLM
Fine-tune to GUI domain
GUI grounding model
Align to novel GUI environment (ours)
Environment-aligned GUI action grounding model
Pre-trained with data of
Various domains
Various tasks
Post-trained with data from K environments:
(x,y) Grounding targets p^t, Screens I^t of various environments, Queries u^t
Post-trained with data from the novel (K+1)th environment:
Screens I^t in a specific environment
GUI-Bee
Actions a^t
Exploration Graph
(x,y) Grounding targets p^t, Screens I^t of a specific environment, Queries u^t including environment knowledge
Input Screen:
Input query: "Reveal layer shadow options".
Models' outputs:
: It is listed in the layer panel. (Failed to ground)
: (0.12, 0.75) (Wrong)
Layers
: (0.22, 0.90) (correct)

Figure 1: We align GUI grounding models from prior works to novel environments. Our proposed alignment process on the top right includes first exploring the specific novel environment with the GUI-Bee agent to generate the exploration graph and then fine-tuning the model with the data from the exploration graph. In the inference example at the bottom, the models encounter a query requiring knowledge of an environment-specific action outcome, which highlights the importance of the proposed alignment process.

Abstract

Graphical User Interface (GUI) action grounding, mapping language instructions to actionable elements on GUI screens, is important for assisting users in interactive tutorials, task automation, accessibility support, etc. Most recent works of GUI action grounding use large GUI datasets to fine-tune Multimodal Large Language Models (MLLMs). However, the fine-tuning data is inherently limited to specific GUI environments, leading to significant performance degradation in novel environments due to the generalization challenges in the GUI domain. Therefore, we argue that GUI action grounding models should be further aligned with novel environments before deployment to optimize their performance. To address this, we first propose GUI-Bee, an MLLM-based autonomous agent, to collect high-quality, environment-specific data through exploration and then continuously fine-tune GUI grounding models with the collected data. To ensure the GUI action grounding models generalize to various screens within the target novel environment after the continuous fine-tuning, we equip GUI-Bee with a novel Q-value-Incentive In-Context Reinforcement Learning (Q-ICRL) algorithm that optimizes exploration efficiency and exploration data quality. In the experiment, we introduce NovelScreenSpot to test how well the data can help align GUI action grounding models to novel environments. Furthermore, we conduct an ablation study to validate the Q-ICRL method in enhancing the efficiency of GUI-Bee. Project page: https://gui-bee.github.io.

[*] Co-advising. This work was partly performed when the first author interned at Adobe Research.

1 Introduction

GUI action grounding maps natural language instructions to specific executable elements or locations on a GUI screen. It is valuable in helping either users or GUI automation agents to locate the target GUI elements to act upon (Agashe et al., 2024; Zheng et al., 2024b; Koh et al., 2024a). As a result, GUI action grounding has become a focus of recent specialized model development efforts (Gou et al., 2024; Cheng et al., 2024; Liu et al., 2024; Chen et al., 2024).

Recent advanced GUI models are mostly fine-tuned from pre-trained MLLMs on data sourced from various GUI domains. However, when facing a novel environment, i.e., one that is not involved during the fine-tuning, these models often face difficulty as the grounding tasks in the novel GUI environment often require environment-specific knowledge that is not generalizable across environments. As illustrated in the inference example at the bottom of Figure 1, a model unfamiliar with the specific GUI environment may fail to infer that a triple-dot icon reveals layer shadow options. Despite the large amount of existing GUI training data, certain environments are inevitably left uncovered. Therefore, we argue that when deployed in novel environments, GUI action grounding models need to be aligned to the novel environments for robust performance.

In this work, we tackle the problem of aligning GUI action grounding models to novel GUI environments that were not included in the previous model training. As a result, our alignment process allows GUI developers to strengthen existing GUI action grounding models for their specific novel use cases. As shown in the top right of Figure 1, the process mainly includes data collection and then model alignment. To realize efficient data collection in any GUI environment, we introduce the GUI-Bee agent. This MLLM-based agent can autonomously explore GUI environments, where it predicts GUI actions and gathers GUI screens after each action is executed. Then, GUI-Bee further autonomously annotates these data from the exploration and further uses them to align GUI action grounding models to the environments explored via fine-tuning.

GUI-Bee adopts an in-context action selection policy that aims to realize a comprehensive exploration of the environment, covering as many novel screens as possible. It leverages a Multimodal Large Language Model (MLLM) with the in-context learning method and a Q-value table and effectively handles the challenge of noisy action space–the environment provided action candidates might be invalid, and unknown action outcome–the execution of a new action could lead to a previously explored screens. A newly proposed Q-value-incentive In-context Reinforcement Learning (Q-ICRL) algorithm is used to optimize the policy dynamically throughout the exploration. The Q-ICRL mainly adjusts the Q-value table based on the exploration history, ensuring the policy prioritizes actions that are less explored and lead to unexplored screens while avoiding repetitive or invalid actions.

To evaluate how GUI-Bee could help align GUI action grounding models to novel environments, we propose the NovelScreenSpot benchmark to evaluate the performance improvements of GUI action grounding models on five novel GUI environments that they are not previously trained on. NovelScreenSpot features human-collected queries requiring rich environment-specific knowledge. In the experiments, we first align models to the five GUI environments by leveraging the GUI-Bee agent to explore the environments and fine-tune the models with the collected data. The results show that models after the alignment significantly outperform their pre-aligned counterparts, confirming the effectiveness of the collected data. Additionally, we perform an ablation study to evaluate the GUI-Bee agent with newly proposed metrics for screen diversity coverage and environment knowledge coverage. Our findings reveal that the Q-ICRL method boosts the efficiency of the exploration for collecting high-quality data.

The overall contributions of this paper are:

- We propose to align GUI action grounding models to novel GUI environments, equipping them with environment-specific knowledge.
- We introduce the GUI-Bee agent with the Q-ICRL algorithm, designed to explore GUI environments and generate high-quality data autonomously.
- We align existing GUI action grounding models to five novel environments using data collected by GUI-Bee and evaluate their performance with the NovelScreenSpot benchmark.
- We propose novel metrics for evaluating the exploration efficiency of the GUI-Bee agent, demonstrating the effectiveness of the Q-ICRL method against baselines.

2 Related Works

2.1 GUI Grounding with MLLMs

As Multimodal Large Language Models advance, recent GUI works have emphasized the visual modality of screen (Deng et al., 2023; Koh et al., 2024a; Xie et al., 2024). GUI action grounding—linking natural language queries to GUI elements—has become a key challenge. Early works (Zheng et al., 2024a; Koh et al., 2024b) used zero-shot MLLMs with SoM methods (Yang et al., 2023), while recent studies focus on generalization. SeeClick (Cheng et al., 2024) introduced visual-only grounding for cross-platform flexibility, and GUICourse (Chen et al., 2024) expanded beyond executable elements. Further advancements address high-resolution, interleaved UI content through improved model designs (Gou et al., 2024; Lin et al., 2024; You et al., 2024) and large-scale in-domain training (Wu et al., 2024; Liu et al., 2024). However, while prior works attempt to generalize GUI grounding models to any GUI environment, they overlook the existence of environment-specific knowledge in GUI action grounding that is hard to generalize. In our work, we instead propose that, given any novel environment, aligning GUI action grounding models to the environment to boost the model performance. Our method can be applied on top of all other GUI grounding models and significantly improve their performance for the deployment need.

2.2 In-Context Learning

In-context learning (ICL) refers to the method of adapting models to new tasks by providing context (Brown, 2020; Chan et al., 2022; Wang et al., 2023). By including examples directly in prompts, ICL allows large language models (LLMs) to generalize to unseen tasks (Garg et al., 2022; Pan, 2023; Wei et al., 2023). Prior works have explored applying ICL to reinforcement learning (RL) either with model training involved (Laskin et al., 2022; Lee et al., 2024; Xu et al., 2022) or by directly leveraging pre-trained LLMs (Krishnamurthy et al., 2024; Monea et al., 2024). We propose the Q-value-incentive In-context Reinforcement Learning (Q-ICRL), which also utilizes pre-trained LLMs but distinguishes itself by using ICL to predict state-action values. Our approach combines the adaptability of LLMs with RL's optimization-driven structure, enabling efficient action selection in the GUI environment exploration.

3 Aligning GUI Action Grounding Models to Novel Environments with GUI-Bee

In this work, we focus on aligning GUI action grounding models to novel GUI environments and ensuring the generalization of models to various screens within the novel environment. To realize this, we propose a GUI-Bee agent, which autonomously collects data enriched with environment-specific knowledge through exploration and data annotation process. Using this data, we continuously fine-tune the GUI action grounding models to boost their performance. The processes of exploration, data annotation, and fine-tuning are detailed in the following sub-sections.

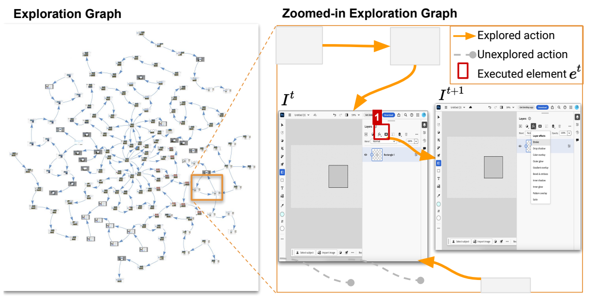


Figure 2: Left: an example of the exploration graph showing screens connected by actions. Right: a zoomed-in view of the graph with examples of I^t and I^{t+1} and some explored and unexplored actions (ever/never selected during the exploration).

3.1 Autonomous Exploration via GUI-Bee

3.1.1 Exploration Goal

The goal of the exploration process is to construct an exploration graph G, where GUI screens I are represented as unique nodes and GUI actions a form the edges connecting these nodes, corresponding to screen transitions. During exploration, GUI-Bee predicts actions to interact with the GUI and captures the screens before and after each action to populate the graph. The process begins from a predefined initial GUI screen I^1 at exploration step 1. At each subsequent step $t, t \in [1, t_{max}]$, where t_{max} is the maximum number of exploration steps, the agent observes the current screen I^t and leverages an MLLM to predict the action a^t. After executing a^t, if I^{t+1} does not exist in the exploration graph G, it will be added to the graph as a new node, and similarly, a^t will be added as a new edge if there is no existing edge in the graph between I^t and I^{t+1}. We detail our method to check if $I^{t+1}$is the same as another screen in G in Appendix B. An example of the exploration graph is shown in Figure 2.

3.1.2 Challenges in the Exploration

Exploration faces two key challenges: identifying valid actions in a noisy action space and handling uncertain screen transitions. The set of action candidates $A_{\text{env}}(I^t)$, obtained from the environment such as a Document Object Model (DOM) tree or preprocessing methods such as OmniParser (Lu et al., 2024), is defined as $a^t \in A_{\text{env}}(I^t) = \{a^t_1, a^t_2, \ldots, a^t_n\}$. However, $A_{\text{env}}(I^t)$ often includes invalid actions targeting non-executable elements, requiring the agent to discern optimal actions that are both valid for exploration. Additionally, screen transitions are unpredictable and often irreversible, complicating decision-making. Effective exploration requires balancing new state discovery with leveraging known actions. This balance necessitates accurate action prediction and robust reasoning, making it critical for the agent to effectively navigate the action space and handle the inherent complexities of GUI environments.

3.1.3 Q-value-Incentive In-Context Reinforcement Learning (Q-ICRL)

Preliminary We consider the exploration process as a Markov Decision Problem, which is defined by a tuple $\langle \mathcal{S}, \mathcal{A}, P, r \rangle$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ represents the action space, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \{0, 1\}$ is the state transition probability function, and $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ denotes the reward function. At each exploration step $t \in \mathbb{N}$, the GUI-Bee agent is at $s^t \in \mathcal{S}$, and leveraging an in-context action selection policy F_θ, GUI-Bee takes an action $a^t \in \mathcal{A}$ on the current observed screen I^t which transitions to a new state $s^{t+1} \in \mathcal{S}$ with probability $P(s^{t+1}|s^t, a^t)$, receiving a reward $r(s^t, a^t)$. The action a consists of the mouse or keyboard movement, e.g., "left click", and the visual of the target element e. The reward r is binary, where it is positive when the a^t leads to a new screen not existing in the exploration graph at the beginning of the current exploration step, i.e., $I^{t+1} \notin G^{t-1}$. Accordingly, to satisfy the Markov property, the state is defined as the exploration graph before the execution of a^t, $s^t = G^{t-1}$, where s^1 contains only the initial screen. For simplicity, s^t is approximated by a set of natural language descriptions $s^t \approx D^t = \{d^k \mid d^k = \text{Describe}(a^k, G^{t-1}), a^k \in G^{t-1}\}$, where a^k represents the edges in the exploration graph and $\text{Describe}(\cdot)$ uses an MLLM to generate descriptions for the actions and screens before and after the action. Further details on this process are provided in Appendix F.

In-Context Action Selection Policy To select an action a^t at state s^t, GUI-Bee adopts an in-context action selection policy $a^t = F_\theta(I_t)$, which takes as input the current screen I^t and outputs an an action $a^t \in A_{\text{env}}(I^t)$ to be executed. F_θ leverages a Q-values table $Q(S, a)$, quantifying the rewards of executing any given actions a at given state S, and a MLLM. To output a^t, F_θ first uses $\{Q(s^t, a^t_i) | a^t_i \in A_{\text{env}}(I^t)\}$ as weights to sample a subset $A'_{\text{env}}(I^t) \subseteq A_{\text{env}}(I^t)$ with length H from the action space $A_{\text{env}}(I^t)$. Then, the MLLM is

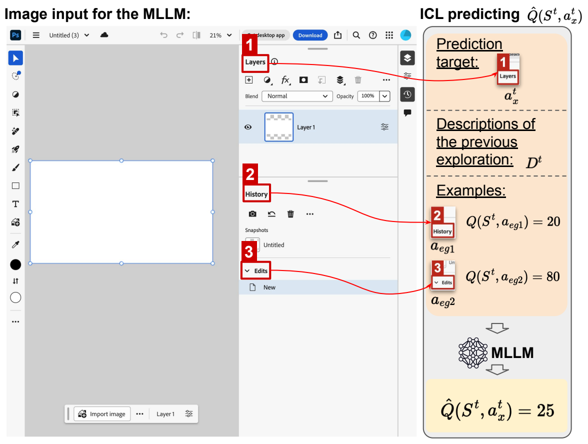


Image input for the MLLM:
ICL predicting $\hat{Q}(S^t, a^t_x)$
1
Layers
2
History
3
Edits
Prediction target:
a^t_x
Descriptions of the previous exploration: D^t
Examples:
$Q(S^t, a_{eg1}) = 20$
a_{eg1}
$Q(S^t, a_{eg2}) = 80$
a_{eg2}
MLLM
$\hat{Q}(S^t, a^t_x) = 25$

Figure 3: Example of predicting the $\hat{Q}(s^t, a^t_x)$ with the MLLM through in-context learning (ICL). Two example actions $(a_{\text{eg1}}, a_{\text{eg2}})$ marked by bounding boxes 2 and 3 are provided as the context along with their Q values. The full prompts are detailed in Appendix C

employed to identify the most promising action a^t from $A'_{\text{env}}(I^t)$ by

$$a^t = argMax_{a^t_x \in A'_{\text{env}}(I^t)}(\hat{Q}(s^t, a^t_x)),$$

where $\hat{Q}(s^t, a^t_x)$ is an MLLM-prediction of the $Q(s^t, a^t_x)$ with the in-contxt learning method. Specifically, to generate $\hat{Q}$, we provide the MLLM with two example actions $(a_{\text{eg1}}, a_{\text{eg2}})$ along with their corresponding Q-values $(Q(s^t, a_{\text{eg1}}), Q(s^t, a_{\text{eg2}}))$ and natural language description of the current state s^t. As the example shown in Figure 3, we mark $(a_{\text{eg1}}, a_{\text{eg2}})$ and a^t_x on the screenshot of I^t as the visual input to the MLLM with the Set-of-Mark method (Yang et al., 2023), and the example is selected to be elements on I^t that are most visually similar to the e^t_x while having more reliable Q-values. We detail the procedure to select example action in Appendix C. Instead of only relying on the $Q(s^t, a^t_i)$ to select a^t, in this way, the internal knowledge of the MLLM helps to offset the potential error in the Q-value table.

Policy Update via Q-ICRL As more screens are included in the exploration graph, the action selection policy F_θ needs to be dynamically updated for optimal efficiency. Intuitively, F_θ will prioritize actions that leads to screens not yet in the exploration graph. We achieve this by using a newly proposed Q-value-Incentive In-Context Reinforcement Learning (Q-ICRL) method, outlined in Algorithm 1, which adjusts the F_θ based on the reward calculated after each exploration step. Inspired by the

Algorithm 1: Q-value-incentive In-context Reinforcement Learning (Q-ICRL)

```
Input: Environment Env, Initial screen $I^1$, Maximum exploration steps $T$
Output: Exploration graph $G$
Initialize $G = \{I^1\}$
for each exploration step $t \in [1, T]$ do
  $s^t = G$
  $a^t \leftarrow F_\theta(I^t)$ // $F_\theta$ is an in-context action selection policy
  $I^{t+1} \leftarrow$ Env.execute($a^t$)
  if $I^{t+1} \notin G$ then
    $G$.add_node($I^{t+1}$)
    $G$.add_edge($I^t, a^t, I^{t+1}$)
  end if
  $\theta \leftarrow \textsc{Update}(\theta, a^t, I^t, A_{\text{env}}(I^{t+1}), I^{t+1})$
  $G \leftarrow G \oplus \{a^t, I^{t+1}\}$
end for
```

Q-learning algorithm (Watkins and Dayan, 1992), the update target is the Q-value table Q that used by F while the MLLM is left frozen. At $t = 1$, the Q-value table is initialed as empty. Then, after the a^t is selected and executed, we first examine the elements on the new screen I^{t+1} and determine the elements exist in the previous explored screens

$$X_i^{t+1} = \{x | a_i^{t+1} \in A_{\text{env}}(I^x), x \in [1, t]\}).$$

Then, the values in the Q-value table are updated via

$$Q(s^{t+1}, a_i^{t+1}) = \begin{cases} Q(s^x, a_i^{t+1}), & \text{if } X_i^{t+1} \neq \varnothing \\ 100, & \text{otherwise} \end{cases}, \quad \text{for } \forall a_i^{t+1} \in A_{\text{env}}(I^{t+1}),$$

where $x = max(X_i^{t+1})$. This mainly propagates any existing non-default value $Q(s^x, a_i^{t+1})$ to the Q values of the new state s^{t+1} and a same action a_i^{t+1}. Further, the Q value for the executed action at current state s_t is updated to reflect the desirability of the action's result by

$$Q(s^t, a^t) = \gamma \cdot \text{Mean}(\{Q(s^{t+1}, a_i^{t+1}) \mid a_i^{t+1} \in A_{\text{env}}(I^{t+1})\})$$

$$\gamma = \begin{cases} \gamma_{\text{max}}, & \text{if } I^{t+1} \notin s^t, \\ \gamma_{\text{med}}, & \text{if } I^{t+1} \in s^t \text{ and } I^{t+1} \neq I^t, \\ \gamma_{\text{low}}, & \text{if } I^{t+1} = I^t, \end{cases}$$

where $\gamma_{\text{max}}, \gamma_{\text{med}}$ and γ_{low} are hyper parameters that we set to be 0.85, 0.75 and 0.4 respectively. This update mechanism ensures that Q-values will decrease for an action after each time it is executed, and when the action leads to new screens with more unseen candidate actions, the decrease is slower than the case if it leads to redundant or ineffective transitions. We further compare our Q-ICRL method with ICRL (Monea et al., 2024) in Appendix G.

3.2 Autonomous Data Annotation with GUI-Bee

After the exploration, for each edge a^t in the G, the connected nodes (I^t, I^{t+1}) is sent to the MLLM to generate u^t, a list of queries serving as action grounding queries for the target element e^t in I^t. We carefully design the prompt, detailed in Appendix E, to guide the MLLM in generating these queries. The queries involve both "what is currently visible" on the screen and "what will appear" after interacting with GUI elements. Building on the multi-lens prompting method (Fan et al., 2024), we create separate visual prompts, or lenses: two capture the full screens of I^t and I^{t+1} while one isolates e^t to ensure high-quality outputs. The center of e^t is then sampled as the target point p^t, and the resulting data (u^t, I^t, p^t) is used to fine-tune GUI action grounding models.

3.3 Fine-tuning Models with Environment-specific Data

We leverage the data generated by the GUI-Bee agent's exploration and annotation processes to fine-tune GUI grounding models to align them with specific novel GUI environments. The data consists of pairs of inputs, including the GUI screen I (comprising a screenshot and an accessibility tree, detailed in Appendix H) and grounding query u, along with corresponding outputs, the target location p. Benefiting from the flexibility of the representation of I in the generated exploration data, we are able to fine-tune models in two input configurations: vision-only and Vision+A11y. In the vision-only configuration, the input consists solely of GUI screenshots. In the Vision+A11y configuration, the input includes both GUI screenshots and the accessibility tree (A11y tree), embedded as part of the text prompt. The fine-tuning process aims to adapt the models to leverage environment-specific knowledge efficiently, improving their grounding performance in the target environments.

		Shopping	Classifieds	Reddit	Eventbrite	Photoshop-web
NovelScreenSpot Benchmark	# Unique Action Grounding Targets	58	42	44	47	44
	# Unique Grounding Queries	105	98	96	107	106
	Ratio of Queries about Action Outcomes	30.5%	32.7%	39.6%	42.1%	34.0%
Exploration Generated Data	# Unique Action Grounding Targets	555	530	590	526	692
	# Unique Grounding Queries	6,080	5,719	6,480	5,740	6,876

Table 1: Statistics of the NovelScreenSpot benchmark and the exploration data generated by GUI-Bee.

4 NovelScreenSpot Benchmark

Overview We introduce NovelScreenSpot, a benchmark with real user grounding queries in five diverse and novel web GUI environments. It simulates real-world GUI model deployment, where GUI action grounding models need to be deployed in environments that are not presented in training data. Therefore, unlike existing benchmarks that emphasize diversity across many environments, such as the ScreenSpot (Cheng et al., 2024), NovelScreenSpot instead includes environments that rarely exist in the training data of the GUI model and provides a greater data variation within each environment.

It consists of triplets: grounding queries, screens (screenshots and accessibility trees), and ground-truth bounding boxes for grounding targets. Table 1 shows benchmark statistics, with examples in Appendix J. Notably, the NovelScreenSpot includes a large number of queries, around one-third of the total benchmark, focusing on interaction outcomes. These queries require environment-specific knowledge and hardly exist in the existing benchmarks. The five GUI environments in the NovelScreenSpot are three offline websites from the VisualWebArena (Koh et al., 2024a)—Shopping, Classifieds (a second-hand marketplace), and Reddit (an online forum)—and two online websites, Photoshop-web and Eventbrite. These environments vary greatly in style, with Photoshop-web dominated by professional icons, Shopping and Classifieds emphasizing images, and Reddit and Eventbrite focusing more on textual content.

Task and Metrics The models are required to predict points within the target GUI elements corresponding to the language queries in NovelScreenSpot, simulating how users indicate a GUI element with a cursor. We evaluate models on NovelScreenSpot before and after continual fine-tuning, quantifying improvements per environment. We define two testing scenarios: *vision+A11y*, where models receive screenshots, queries, and accessibility tree text, and *vision-only*, where inputs are limited to screenshots and queries. A prediction is correct if it falls within the ground-truth bounding box, with accuracy as the evaluation metric.

Annotation NovelScreenSpot is manually constructed through a multi-step annotation process. First, we ask annotators to interact with the web environments and record their actions, including the screens before and after the action and the corresponding target elements. Next, different annotators write queries based on three perspectives: direct element name/label, element appearance, and interaction outcome. Finally, we manually validate data by eliminating ambiguous queries, removing duplicates, and discarding misaligned annotations. The resulting dataset provides clear triplets of queries, screens, and target elements. An example annotation interface is in Appendix I.

5 Experiments

5.1 Alignment Effectiveness

We evaluate the effectiveness of GUI-Bee data in aligning GUI action grounding models to novel environments. We fine-tune four models—SeeClick (Cheng et al., 2024), Qwen-GUI (Chen et al., 2024), UIX-7B (Liu et al., 2024), and Qwen2.5-VL 3B (Yang et al., 2024b)—to adapt to previously unseen GUI environments in the NovelScreenSpot benchmark.

Setups We employ the GUI-Bee agent to explore the five environments in NovelScreenSpot, conducting up to 400 exploration steps per environment with three candidate actions sampled at each step. To ensure diverse screen data, the exploration is repeated three times per environment at varying screen resolutions. GPT-4o (OpenAI, 2024) is adopted as the multimodal large language model (MLLM) in the experiments, though other MLLMs can also be integrated into the framework. Table

	NovelScreenSpot						Multimodal-M2W
	Shopping	**Classifieds**	**Reddit**	**Eventbrite**	**Photoshop-web**	**Avg.**	**Eventbrite**
			Vision-only GUI Action Grounding				
SeeClick	36.2	36.7	35.4	35.5	13.2	-	23.1
SeeClick$_{\text{Mind2Web}}$	39.0 *(+2.8)*	30.6 *(-6.1)*	36.5 *(+1.1)*	43.0 *(+7.5)*	9.4 *(-3.8)*	*(+0.3)*	38.5 *(+15.4)*
SeeClick$_{\text{GUI-Bee}}$(Ours)	48.6 *(+12.4)*	44.9 *(+8.2)*	39.6 *(+4.2)*	53.3 *(+17.8)*	18.9 *(+5.7)*	*(+9.7)*	38.5 *(+15.4)*
UIX-7B	31.4	38.8	44.8	43.0	22.6	-	23.1
UIX-7B$_{\text{Mind2Web}}$	39.0 *(+7.6)*	38.8 *(+0)*	37.5 *(+7.3)*	43.9 *(+0.9)*	16.0 *(-6.6)*	*(+1.8)*	23.1 *(+0)*
UIX-7B$_{\text{GUI-Bee}}$(Ours)	78.1 *(+46.7)*	66.3 *(+27.5)*	60.4 *(+15.6)*	70.1 *(+27.1)*	31.1 *(+8.5)*	*(+25.1)*	**53.8** *(+30.7)*
Qwen-GUI	19.0	21.4	26.0	34.6	9.4	-	23.1
Qwen-GUI$_{\text{Mind2Web}}$	23.1 *(+4.1)*	21.6 *(+0.2)*	28.1 *(+2.1)*	36.0 *(+1.4)*	10.3 *(+0.9)*	*(+1.7)*	23.1 *(+0)*
Qwen-GUI$_{\text{GUI-Bee}}$(Ours)	25.7 *(+6.7)*	31.6 *(+10.2)*	28.1 *(+2.1)*	37.4 *(+2.8)*	12.3 *(+2.9)*	*(+4.9)*	46.2 *(+23.1)*
Qwen2.5-VL	70.5	57.1	62.5	71.0	43.4	-	46.2
Qwen2.5-VL$_{\text{Mind2Web}}$	68.8 *(-1.7)*	43.4 *(-13.7)*	55.2 *(-7.3)*	75.7 *(+4.7)*	31.6 *(-11.8)*	*(+1.7)*	46.2 *(+0)*
Qwen2.5-VL$_{\text{GUI-Bee}}$(Ours)	**75.0** *(+4.5)*	**73.1** *(+16.0)*	**79.5** *(+17.0)*	**78.6** *(+7.6)*	**56.3** *(+12.9)*	*(+11.6)*	**53.8** *(+7.6)*
			Vision+A11y GUI Action Grounding				
Qwen-GUI	34.3	50.0	34.4	52.3	13.2	-	-
Qwen-GUI$_{\text{GUI-Bee}}$(Ours)	51.4 *(+17.1)*	54.1 *(+4.1)*	55.2 *(+20.8)*	62.6 *(+10.3)*	41.5 *(+28.3)*	*(+16.1)*	-
UIX-7B	16.2	14.3	11.5	21.5	10.4	-	-
UIX-7B$_{\text{GUI-Bee}}$(Ours)	**74.3** *(+58.1)*	77.6 *(+63.3)*	80.2 *(+68.7)*	82.2 *(+60.7)*	**70.8** *(+60.4)*	*(+62.2)*	-
Qwen2.5-VL	67.6	62.3	57.3	65.4	40.6	-	-
Qwen2.5-VL$_{\text{GUI-Bee}}$(Ours)	72.3 *(+4.7)*	**78.8** *(+16.5)*	**83.3** *(+26.0)*	**74.8** *(+9.4)*	59.8 *(+19.2)*	*(+15.2)*	-

Table 2: Results of benchmarking GUI grounding models. We show the model accuracy and the absolute improvement over the vanilla models after the models are aligned to the environment. The results demonstrate that our GUI-Bee model significantly improves the performance of GUI action grounding models in novel environments.

1 summarizes the exploration statistics, with costs under $50 per environment. For Qwen-GUI, UIX-7B, and Qwen2.5-VL, we further incorporate both vision-only and Vision+A11y data, consistent with their pre-fine-tuning formats. Additional details on exploration settings, data formatting, and fine-tuning are provided in Appendix H.

Main Results In Table 2, we report performance on the NovelScreenSpot benchmark before and after alignment to each novel GUI environment. For each environment, we evaluate two adaptation strategies: (1) fine-tuning with a broad, general-purpose dataset from Multimodal-Mind2Web (Zheng et al., 2024b), and (2) fine-tuning with the environment-specific data collected by GUI-Bee. This setup simulates a practical developer choice when deploying to a new application: either re-use existing general GUI grounding data or leverage targeted data gathered via GUI-Bee. The results show that GUI-Bee-collected data shows the best effectiveness in improving the performance of all tested models, underscoring the value of environment-specific knowledge for adaptation. Among the models, UIX-7B achieves the largest relative accuracy gains, while Qwen2.5-VL reaches the highest absolute accuracy after alignment.

Furthermore, models fine-tuned in the *vision+A11y* GUI action grounding setting demonstrate greater performance for some heavy-text or dense-icon environments such as classifieds, Reddit, and Photoshop-web. This underscores the value of the flexible formats in GUI-Bee's collected data, which incorporate contextual information such as accessibility data to enhance model performance.

Results on Grounding for Offline GUI Agents We extend our evaluation by following Gou et al. (2024) to leverage the test data from Multimodal-Mind2Web (Zheng et al., 2024b) for further assessing the GUI grounding models after the alignment. This test data simulates GUI grounding tasks from MLLM-planned instructions in GUI agent applications, corresponding to high-level, real-world GUI tasks. The grounding queries consist of element descriptions generated by MLLMs, without explicitly referencing coordinates. As shown in Table 2, the result reveals that models continuously fine-tuned with GUI-Bee data achieve significantly greater performance improvements in the target environment. This further validates the effectiveness of our proposed method.

Model Performance on Action Outcomes Related Queries In Figure 4, we present the average performance improvements of the models on queries related to action outcomes that require strong environment-specific knowledge, as well

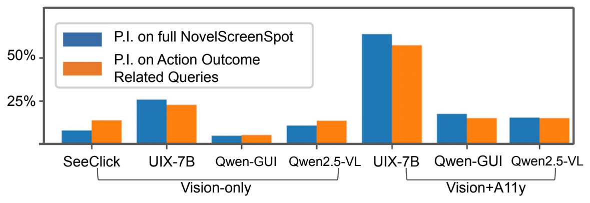


Figure 4: Model average performance improvements (P.I.) on the full NovelScreenSpot benchmark v.s. on the subset with queries related to action outcome. The proposed alignment improves model performance evenly.

as their overall performance improvements after the proposed alignment process. The results show consistent gains in both categories across all models. This demonstrates that the data generated by our GUI-Bee agent is universally effective in enhancing model performance for grounding tasks in the novel GUI environment, addressing not only environment-specific challenges but also general grounding weaknesses faced by models.

5.2 Exploration Efficiency

To enable a successful alignment of the GUI action grounding model to a novel environment, it is critical to efficiently collect diverse data specific to the environment rather than repeating data (an ablation study is shown in Appendix D). To further evaluate the efficiency of our GUI-Bee agent in exploring and generating diverse data within GUI environments, we compare it against two baseline exploration methods on the three offline GUI environments: Shopping, Classifieds, and Reddit. These environments are reset to identical initial states at the beginning of each exploration, ensuring that all agents start from the same conditions and face equivalent challenges.

Evaluation and Metrics To evaluate the efficiency of exploring diverse data, we assess the diversity of actions and screens in the exploration graph $G^{t_{max}}$ generated by each agent under the same maximum number of exploration steps t_{max}. We introduce the Depth-fixed DOM Diversity Counts (D3C) metric to assess structural variation in the screens within the exploration graph G^t generated by different agents objectively. D3C is defined as the number of distinct page structures in the G^t. Each page structure is determined by truncating the DOM tree of a screen to a fixed depth, retaining only the class attributes of elements. By counting the unique page structures within all the page structures within G^t, we get the D3C value

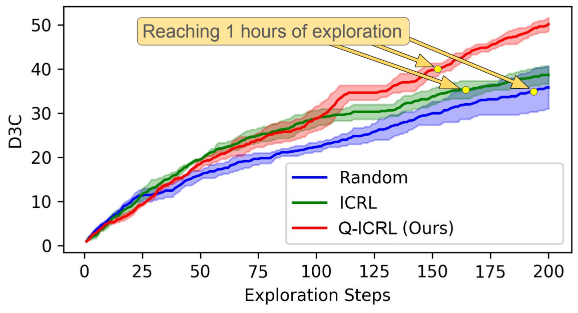


Figure 5: Mean and standard deviation of Depth-fixed DOM Diversity Counts (D3C) at various exploration steps across three runs in three environments. GUI-Bee agent demonstrates a wider exploration coverage with the same exploration steps and time costs.

at the exploration step t. With a fixed number of exploration steps, D3C provides a quantitative measure of the agent's efficiency in uncovering diverse structural layouts, offering a clear and objective metric for exploration breadth.

Baselines To evaluate the Q-ICRL method, we introduce two ablated versions of GUI-Bee: one explores with the In-Context Reinforcement Learning (ICRL) method (Monea et al., 2024) and the other one explores with the random strategy. The ICRL version of our GUI-Bee ablates the use of the Q-value table, and it directly leverages the MLLM with in-context learning to select the next action on the GUI screen. The in-context example is randomly chosen from the actions that exist in the exploration graph. The random agent follows a purely stochastic strategy, selecting actions randomly from the candidate actions of each screen. All agents are constrained to the same maximum number of action steps to enable a fair comparison.

Results We calculate the D3C for each exploration conducted by agents in all three environments and compute the average D3C for each agent across the three environments. This process is repeated three times, and the mean and standard deviation of the averaged D3C across these evaluations are plotted in Figure 5. As the number of exploration steps increases, the averaged D3C for all agents grows, but our GUI-Bee agent demonstrates stronger growth momentum and significantly outperforms the baselines after 100 exploration steps. Additionally, we analyze the time costs of the exploration that we experiment with and pinpoint the time step that corresponds to a time cost of one hour after the exploration starts for each exploration method in Figure 5. We find that the Q-ICRL achieves a higher D3C within the same amount of time and the exploration with Q-ICRL gets a D3C of 35 using only 50% of the exploration steps and around 72% of the total time compared to random exploration. More details about the exploration time costs are in Appendix H. These results collectively demonstrate that the GUI-Bee agent is more efficient in covering broader areas of the GUI environment and uncovering more diverse exploration data compared to the baseline.

6 Conclusion

This work pioneers the alignment of GUI action grounding models to novel environments. The alignment leverages the GUI-Bee agent, a newly proposed MLLM-based agent with a Q-ICRL algorithm, to first autonomously explore GUI environments and collect data. Then, using the data, GUI action grounding models are continuously fine-tuned. Our experiments show significant performance gains across all target novel environments and demonstrate GUI-Bee's efficiency in generating diverse environment-specific data. Additionally, there is a strong potential for extending GUI-Bee toward collecting data for multi-step navigation tasks. Such an extension could support applications like workflow automation, complex web browsing, and creative tool guidance. Realizing this vision raises important open challenges, such as designing Q-value update mechanisms that ensure semantically consistent trajectories and scaling exploration to longer-horizon tasks, which we identify as promising directions for future research.

7 Limitations

The GUI-Bee agent excels in tailoring GUI action grounding models for specific environments but has limitations. It uses Multimodal Large Language Models (MLLMs) like GPT-4o, which provide high-quality data but come with higher computational costs, latency, and privacy issues. Additionally, as exploration within a specific environment continues and the number of exploration steps increases, the GUI-Bee agent accumulates an increasingly long history of exploration data. Processing this longer history can introduce additional overhead, increasing the time required to select actions for subsequent exploration steps. This may impact scalability in highly complex environments or during exploration with excessive number of exploration steps.

References

Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Eric Wang. 2024. Agent s: An open agentic framework that uses computers like a human. *arXiv preprint arXiv:2410.08164*.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*.

Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Stephanie Chan, Adam Santoro, Andrew Lampinen, Jane Wang, Aaditya Singh, Pierre Richemond, James McClelland, and Felix Hill. 2022. Data distributional properties drive emergent in-context learning in transformers. *Advances in Neural Information Processing Systems*, 35:18878–18891.

Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, Yuan Yao, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2024. Guicourse: From general vision language models to versatile gui agents.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. 2024. Seeclick: Harnessing gui grounding for advanced visual gui agents. *arXiv preprint arXiv:2401.10935*.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yue Fan, Lei Ding, Ching-Chen Kuo, Shan Jiang, Yang Zhao, Xinze Guan, Jie Yang, Yi Zhang, and Xin Wang. 2024. Read anywhere pointed: Layout-aware gui screen reading with tree-of-lens grounding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9503–9522.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. 2022. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598.

Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. 2024. Navigating the digital world as humans do: Universal visual grounding for gui agents. *arXiv preprint arXiv:2410.05243*.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. 2024a. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. *arXiv preprint arXiv:2401.13649*.

Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. 2024b. Tree search for language model agents. *arXiv preprint arXiv:2407.01476*.

Akshay Krishnamurthy, Keegan Harris, Dylan J Foster, Cyril Zhang, and Aleksandrs Slivkins. 2024. Can large language models explore in-context? *arXiv preprint arXiv:2403.15371*.

Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Hansen, Angelos Filos, Ethan Brooks, and 1 others. 2022. In-context reinforcement learning with algorithm distillation. *arXiv preprint arXiv:2210.14215*.

Jonathan Lee, Annie Xie, Aldo Pacchiano, Yash Chandak, Chelsea Finn, Ofir Nachum, and Emma Brunskill. 2024. Supervised pretraining can learn in-context reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Bo Li, Kaichen Zhang, Hao Zhang, Dong Guo, Renrui Zhang, Feng Li, Yuanhan Zhang, Ziwei Liu, and Chunyuan Li. 2024. Llava-next: Stronger llms supercharge multimodal capabilities in the wild.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Weixian Lei, Lijuan Wang, and Mike Zheng Shou. 2024. Showui: One vision-language-action model for gui visual agent. *arXiv preprint arXiv:2411.17465*.

Junpeng Liu, Tianyue Ou, Yifan Song, Yuxiao Qu, Wai Lam, Chenyan Xiong, Wenhu Chen, Graham Neubig, and Xiang Yue. 2024. Harnessing webpage uis for text-rich visual understanding. *arXiv preprint arXiv:2410.13824*.

Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. 2024. Omniparser for pure vision based gui agent. *Preprint*, arXiv:2408.00203.

Giovanni Monea, Antoine Bosselut, Kianté Brantley, and Yoav Artzi. 2024. Llms are in-context reinforcement learners. *arXiv preprint arXiv:2410.05362*.

OpenAI. 2024. Gpt-4o.

Jane Pan. 2023. What in-context learning "learns" in-context: Disentangling task recognition and task learning. Master's thesis, Princeton University.

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023. Label words are anchors: An information flow perspective for understanding in-context learning. *arXiv preprint arXiv:2305.14160*.

Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning*, 8:279–292.

Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, and 1 others. 2023. Larger language models do in-context learning differently. *arXiv preprint arXiv:2303.03846*.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and 1 others. 2024. Os-atlas: A foundation action model for generalist gui agents. *arXiv preprint arXiv:2410.23218*.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Preprint*, arXiv:2404.07972.

Mengdi Xu, Yikang Shen, Shun Zhang, Yuchen Lu, Ding Zhao, Joshua Tenenbaum, and Chuang Gan. 2022. Prompting decision transformer for few-shot policy generalization. In *international conference on machine learning*, pages 24631–24645. PMLR.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 40 others. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024b. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. 2023. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *arXiv preprint arXiv:2310.11441*.

Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei Yang, and Zhe Gan. 2024. Ferret-ui: Grounded mobile ui understanding with multimodal llms. *arXiv preprint arXiv:2404.05719*.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. 2024a. Gpt-4v (ision) is a generalist web agent, if grounded. *arXiv preprint arXiv:2401.01614*.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. 2024b. Gpt-4v(ision) is a generalist web agent, if grounded. *arXiv preprint arXiv:2401.01614*.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, and 1 others. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

A GUI Elements Fuzzy Visual Matching

We develop a GUI Elements Fuzzy Visual Matching module F_{fvm}, to compare if two GUI elements, e and e', can be recognized as visually the same. Challenges arise from the variations in GUI rendering; for example, web browsers could render the sample page with slight element shifts each time. Such variation can make pixel-perfect matching overly sensitive, leading to false negatives. Furthermore, dynamic elements on the screen, such as GIFs, can also cause variability unrelated to the executed action. We first let F_{fvm} apply a Gaussian filter to e and e' to smooth rendering defects. Then we enumerate an offset value o for e and e' to overlay e on e' and compute normalized pixel-wise difference $F_{fvm}(e, e')$, while ensuring a minimum overlap of 75%, IoU$(e, e') \geq 0.75$. Finally, from all possible offset values, e and e' are considered identical if based on the maximum of $F_{fvm}(e, e')$ is less equal to 0.05:

$$max_{o(F_{fvm}(\text{Shift}(e,o),e'))} \leq 0.05$$

B Determining the Equality of Two Screens

We leverage the proposed GUI Elements Fuzzy Visual Matching module detailed in Appendix A to verify whether a screen I is the same as another screen I'. Specifically, we compute the $max_{o(F_{fvm}(\text{Shift}(e,o),e'))}$ for all e in I and e' cropped at the same location in I'. If the $max_e(max_{o(F_{fvm}(\text{Shift}(e,o),e'))}) \leq 0.05, \forall e \in I^t, \forall e' \in T'$, we regard the I and I' are the same. Additionally, if dynamic content exists in screen I, we capture the screenshot for the same screen multiple times over time to identify inconsistent regions and excludes them when determining the equality for I and any other screen I'.

C Details of In-Context Action Selection Policy

As introduced in Section 3.1.3, the in-context action selection policy uses the MLLM to predict $\hat{Q}(s^t, a^t_x)$ and thus select the action a^t to execute next at time step t. For a candidate action a^t_x, to enable the MLLM to correctly estimate its Q-value reflecting its desirability at the current state, we provide the MLLM with example pairs $(a_{\text{eg1}}, Q(s^t, a_{\text{eg1}}))$ and $(a_{\text{eg2}}, Q(s^t, a_{\text{eg2}}))$. The example actions are chosen from the same candidate set $A_{\text{env}}(I^t)$ as a^t_x, with $(Q(s^t, a_{\text{eg1}}), Q(s^t, a_{\text{eg2}})) \neq 100$, and their corresponding elements $(e_{\text{eg1}}, e_{\text{eg2}})$ being the most visually similar to e^t_x. This ensures that the example actions either correspond to or are closely related to previously executed actions and are visually close to the candidate action a^t_x. To identify such visually similar elements, we develop a GUI Element Fuzzy Visual Matching module (Appendix A). If no suitable a_{eg} exists, we skip the example-related content in the prompt.

To construct the text prompt for predicting $\hat{Q}(s^t, a^t_x)$, we include the natural language descriptions D^t approximating the current state s^t, the two example actions and their Q-values, and the visual input encoded using the Set-of-Mark method (Yang et al., 2023). Figure 10 shows an example of the full prompt input and the corresponding model output.

D Ablation Study on Data Diversity and Scale

Setup and Metrics. We study how *diversity* and *scale* of environment-specific data affect alignment. We quantify structural diversity with the Depth-fixed DOM Diversity Counts (D3C) score (higher is more diverse), and we report downstream grounding accuracy for multiple backbones under two settings: (i) **Vision-only** grounding and (ii) **Vision+A11y** grounding.

Model Alignment with Data of Various Diversity. We test the hypothesis that more diverse environment-specific data collected by GUI-Bee leads to stronger alignment. Concretely, we ask: *If the same number of samples is gathered with a naive random-action explorer—hence much lower diversity—does the aligned model still improve?* On the Photoshop environment of NovelScreenSpot, we fine-tune four GUI grounding models on equal-sized datasets collected by (a) uniform-random exploration (low D3C) and (b) our Q-ICRL explorer (high D3C). As shown in Table 3, all models improve more with Q-ICRL data, confirming that higher screen diversity yields better alignment.

Model Alignment with Data of Various Scale. We next investigate how scaling the amount of collected data affects alignment when data diversity is kept mostly unchanged. This happens when the exploration within one environment is largely saturated where the newly explored screens become

Model	Before alignment	Random Exploration (low D3C)	Q-ICRL (high D3C)
Vision-only GUI Action Grounding			
SeeClick	13.2	14.6 (+1.4)	**18.9** (+5.7)
Qwen-GUI	9.4	9.5 (+0.1)	**12.3** (+2.9)
UIX-7B	22.6	29.6 (+7.0)	**31.1** (+8.5)
Qwen2.5-VL	43.4	50.0 (+6.6)	**56.3** (+12.9)
Vision+A11y GUI Action Grounding			
Qwen-GUI	13.2	33.3 (+20.1)	**41.5** (+28.3)
UIX-7B	10.4	65.7 (+55.3)	**70.8** (+60.4)
Qwen2.5-VL	40.6	51.9 (+11.3)	**59.8** (+19.2)

Table 3: Impact of data diversity on alignment (equal data size). The results show the performance of four GUI action-grounding models after fine-tuning with data collected either by our GUI-Bee with Q-ICRL algorithm with a higher D3C, or by uniform-random exploration with low D3C. GUI-Bee with Q-ICRL discovers more *diverse* screens (higher D3C) and yields larger gains than random exploration.

Model	Without FT	FT @ 6k	FT @ 15k
Vision-only GUI Action Grounding			
SeeClick	36.2	48.6 (+12.4)	**52.7** (+16.5)
Qwen-GUI	19.0	25.7 (+6.7)	**29.3** (+10.3)
UIX-7B	31.4	**78.1** (+46.7)	76.0 (+44.6)
Qwen2.5-VL	70.5	75.0 (+4.5)	**75.7** (+5.2)
Vision+A11y GUI Action Grounding			
Qwen-GUI	34.3	51.4 (+17.1)	**53.8** (+19.5)
UIX-7B	16.2	74.3 (+58.1)	**82.1** (+65.9)
Qwen2.5-VL	67.6	72.3 (+4.7)	**77.5** (+9.9)

Table 4: Scaling with near-constant structural diversity (D3C $\approx$ constant). Increasing data from 6k to 15k samples (via multi-resolution re-exploration) yields smaller gains than diversity-driven scaling; mild regression can occur due to redundancy.

mainly screens that are non-identical to previously explored ones but structurally the same. To this end, we repeatedly explore the WebArena–Shopping environment six additional times (maximum T=400 steps each) at different screen resolutions. This procedure expands the training set from 6,080 to 18,170 samples, while the Depth-fixed DOM Diversity Counts (D3C) increases only slightly from 79 to 80, i.e., the added data consists mostly of *structurally similar but non-identical* screens. Fine-tuning with this extended dataset still improves overall performance (Table 4), but the gains are smaller than those obtained from diversity-driven scaling. In some cases, such as UIX-7B under the Vision-only setting, redundancy even leads to mild regression. These results highlight that increasing data scale remains beneficial but yields diminishing returns once diversity has plateaued, suggesting that effective exploration should prioritize discovering structurally novel screens over merely accumulating more examples.

Key Takeaways. (i) **Diversity first:** For a fixed budget, collect screens that increase D3C; this consistently yields larger downstream gains. (ii) **Plateau detection:** When D3C growth stalls, further scale (e.g., resolution variants) provides smaller benefits and can slightly regress some models—a signal to stop or switch environments. (iii) **Practical guidance:** Track "new screens per 100 actions" and D3C; terminate exploration when both flatten to avoid redundant data collection.

E GUI Action Grounding Queries Annotation

Once a new edge (I^t, a^t, I^{t+1}) is added during exploration, we send this information to the MLLM to generate u^t, a list of action grounding queries for the target element e^t in I^t. The generation process uses a carefully crafted prompt, designed to ensure the queries cover both queries focused on current screen content and queries anticipating interaction outcomes grounding challenges. The full version of the text prompt for the MLLM is provided in Figure 6. We also show an example of the input images in Figure 9 along with the GUI action grounding queries u^t in the corresponding output.

```
A user clicks the element marked with box 1 on the screen shown in the first image and then arrives at the screen shown in the second image. A zoomed-in look of the element clicked is shown in the third image. Please generate a JSON dictionary format with values for the following 3 keys:
1. analysis: describing the appearance of the element, including but not limited to color, shape, etc. Try to make the description uniquely identify the element.
2. system_1_queries: a list of maximum 6 requests or questions that will uniquely lead to clicking the element in the page, with maximum 3 of them mentioning something special about the appearance of the element (can be skipped if the element's appearance is just plain text).
3. system_2_queries: a list of maximum 5 simple requests or questions that uniquely lead to the element. They each should mention one specific function (consequence) of this click, i.e. a specific thing that is only shown in the second image, but not in the first image.
```

Figure 6: Text prompt for generating GUI action grounding queries (u^t).

F Approximating State (s^t) with Natural Language Descriptions D^t

To simplify the representation of the state s^t at the t-th exploration step, we approximate it with a list of natural language descriptions D^t, where each description d^k corresponds to an action and its resulting state transition. Figure 7 illustrates the input prompt used to generate one such natural language description. The input consists of the current screen I^t with the action target a^t visually marked (box 1), along with the resulting screen I^{t+1}. Using this input, the MLLM produces a textual description capturing the key details of the transition, including the action a^t, the visual changes between I^t and I^{t+1}, and any notable observations. These natural language descriptions serve as a compact and interpretable representation of the exploration history, enabling efficient input to the MLLM during subsequent steps of the Q-ICRL process.

A use clicked an element on the screen shown in the first image, and then arrived at the screen shown in the second image. Your output should be a json dictionary format with values for the following 2 keys:
1. consequence: what happens after the click and what is shown based on the second image.
2. clicked_element: describe what element (marked by the box 1) is clicked (apperance, layout, etc).
Note: box 1 is the bounding box with label 1. Note: do not mention box 1 in your output.

Figure 7: Text prompt for generating a natural language description d^t of one exploration step at t, where the input images are I^t with box 1 marked and I^{t+1}.

G Distinction Between Q-ICRL and ICRL

In recent work, Monea et al. (2024) introduced In-Context Reinforcement Learning (ICRL), demonstrating that Large Language Models (LLMs) can exhibit reinforcement learning-like behaviors without parameter updates. In ICRL, the model's decisions are guided by a context that includes its past predictions and the rewards they generated. While powerful, this approach is distinct from classical reinforcement learning methods that optimize a value function. Our Q-ICRL method builds directly upon the foundation of ICRL but integrates a core principle from Q-learning: learning and optimizing an action-value function (the Q-function). In our framework, the LLM itself serves as the Q-function. During exploration, we dynamically update the in-context memory with trajectories of actions and their observed rewards. This process effectively updates the "Q-values" accessible to the model in-context, allowing it to refine its policy by favoring actions that have led to higher rewards in the past. This dynamic, value-based adjustment of the context is the key differentiator from ICRL, as it enables a more structured and efficient exploration behavior that improves as more interactions are observed.

H Experiment Details for Exploration and Fine-tuning

Exploration Details and Configurations We adopt the WebArena environment (Zhou et al., 2023) for exploration, which is built on Playwright and Chromium. At each screen, the environment directly provides screenshot, DOM tree, and accessibility (A11y) tree for the visible area. During the exploration, GUI-Bee agent operates with a maximum of $T = 400$ exploration steps, and at each step, it samples $H = 3$ candidate actions. The actions in the exploration are restricted to "click" and "scroll" categories, as these are the most common actions for GUI navigation. For "scroll" actions, the target element e^t is simplified to represent the "full page", ensuring consistent representation of scroll transitions. To enhance robustness, each environment is explored three times using different screen resolutions. This variation ensures the generated data captures diverse screen setups, improving the generalization ability of the fine-tuned models. Each exploration starting with empty G lasts between 5 to 18 hours, depending on the computing power of the computer used and the web loading latency. The long loading time is due to the overhead of web loading time, around 3 seconds, and the time using the Playwright tool [*] to acquire the accessibility tree for each screen, around 5 seconds. Other than that, although our Q-ICRL introduces some computation expenses for each exploration step, they are easily parallelizable and on average takes around 3 seconds when a new screen is explored and around 1 second otherwise.

Model Fine-tuning Configurations We fine-tune three GUI grounding models—SeeClick (Cheng et al., 2024), Qwen-GUI (Chen et al., 2024), and UIX-7B (Liu et al., 2024)—using the data generated by the GUI-Bee agent. Fine-tuning is performed in two input configurations: vision-only, where the input consists of GUI screenshots only, and Vision+A11y, where the input includes both GUI screenshots and the accessibility tree embedded in the text prompt.

The accessibility tree (A11y tree) is a structured representation of the GUI that exposes key information about screen elements, such as their type,

[*] https://playwright.dev

properties, and hierarchical relationships. Typically used for assistive technologies like screen readers, the A11y tree provides textual descriptions and spatial information of the interface components, complementing visual input for models. Including this information in the input prompt allows models to leverage both visual and structural cues, improving their grounding accuracy.

SeeClick and Qwen-GUI are based on Qwen-VL (Bai et al., 2023), while UIX-7B is derived from Llava-1.6 (Li et al., 2024) with Qwen2-7B-Instruct (Yang et al., 2024a) as the primary LLM backbone. For models that predict bounding boxes, such as Qwen-GUI and UIX-7B, the center of the predicted bounding box is used as the final output point for evaluation.

Fine-tuning Settings For all models, fine-tuning is conducted with a batch size of 16, a learning rate of 1×10^{-6}, and for 5 training epochs. The generated exploration data are formatted to match the original training format of these models to ensure consistency. For models trained with bounding boxes, the ground truth bounding box coordinates are converted to center points to align with evaluation requirements.

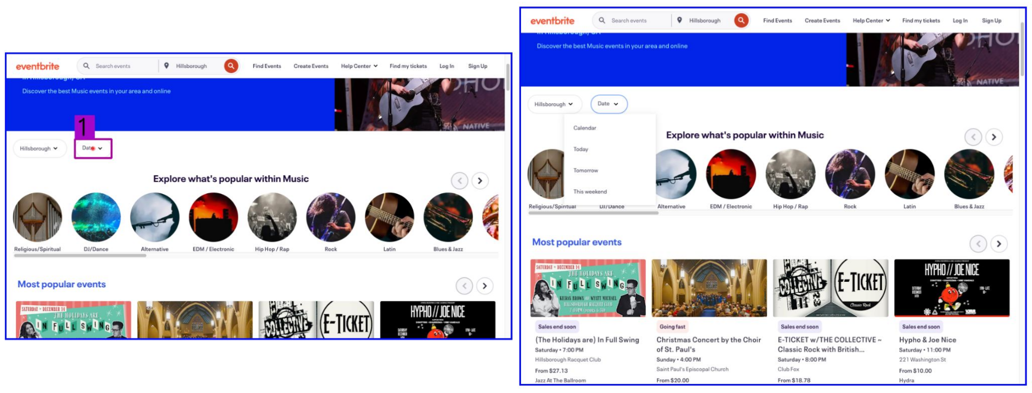
I Human Data Annotation Details

We recruited annotators from within our research team, ensuring familiarity with GUI environments. Annotators were compensated fairly for their work to maintain ethical standards. All queries undergo manual review to eliminate ambiguity, duplication, and low quality, ensuring alignment with recorded actions.

Annotation Interface Figure 8 shows an example of the annotation interface, where annotators view the screens and marked target element to input queries efficiently.

J Examples of NovelScreenSpot

We randomly sample data from each environment in the NovelScreenSpot benchmark and present examples in Figure 11, 12, 13, 14, and 15. Each figure illustrates the GUI screen, the A11y string, the corresponding query, and the ground truth target element, showcasing the diversity and environment-specific nature of the benchmark.



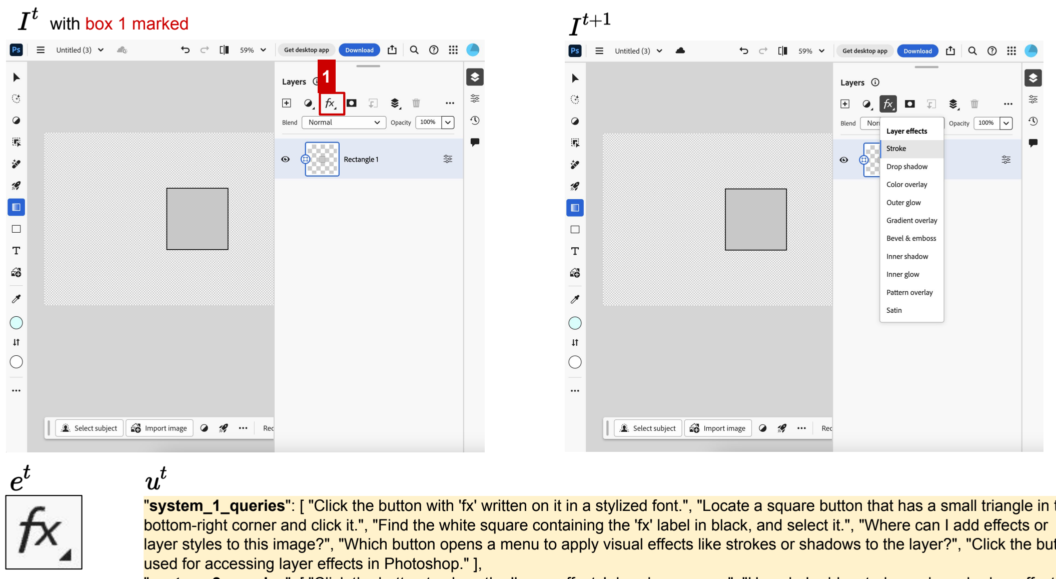
Your task is to write natural language queries or questions that uniquely lead to the action of clicking the target element marked with bounding box 1 in the left image. The right image shows the screen after the action. Each query must be clear, concise, and unambiguous, ensuring it precisely identifies the target element and the action to be performed. You may leave blank if no good answer.

Provide a query/question with the direct name or label of the target element:
____

Provide a query/question with the appearance of the target element, such as its color, shape, or position:
____

Provide a query/question with the outcome of interacting with the target element:
____

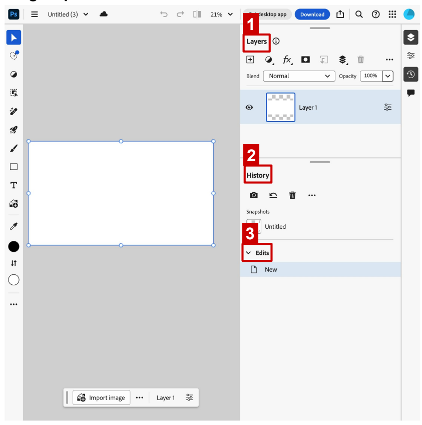
Figure 8: Example of the annotation interface for collecting GUI action grounding queries. The target element is marked with bounding box 1, and annotators will write queries uniquely identifying this action.



"system_1_queries": ["Click the button with 'fx' written on it in a stylized font.", "Locate a square button that has a small triangle in the bottom-right corner and click it.", "Find the white square containing the 'fx' label in black, and select it.", "Where can I add effects or layer styles to this image?", "Which button opens a menu to apply visual effects like strokes or shadows to the layer?", "Click the button used for accessing layer effects in Photoshop."],
"system_2_queries": ["Click the button to show the 'Layer effects' dropdown menu.", "How do I add a stroke or drop shadow effect to this layer?", "Open the menu that lets me choose effects like gradient overlay or bevel & emboss.","Where can I find options like 'Color overlay' or 'Outer glow'?", "Show me the effects I can apply to the layer, such as patterns or shadows."]

Figure 9: Examples of the input images (I^t with box 1 marked, I^{t+1} and e^t) and output GUI action grounding queries u^t in process of generating data generation along the exploration.

Image input for the MLLM:



Prompt for the MLLM:

The first image shows the screenshot of the <u>Photoshop-web</u> website with a some elements marked with bounding boxes. You are an agent exploring the website and the followings are descriptions of the previous exploration:

<u>Previous exploration covers:</u>
<u>1. Trash Bin Icon: Located in the Layers section, indicating a Delete Layer button.</u>
<u>2. Edit drop down button: Positioned in the History panel. Clicking it expands the past edits.</u>

Your task is to generate a rating between 0 and 100 for the element in box 1, where the higher the rating means the more likely the element is: 1. clickable. 2. leading to something looks mysterious or not explored in the previous exploration. 3. leading to something that has been explored but is a must-click to more possibilities (get score no more than 60). Please output a json dictionary with 2 keys: "analysis" and "score".

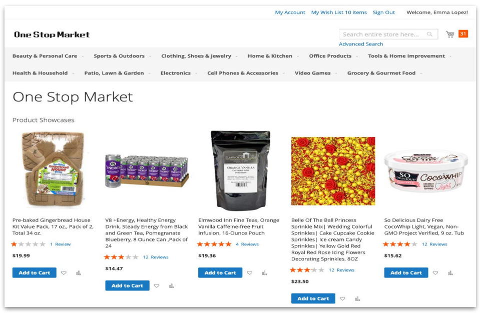
The elements in box 2 and 3 serve as examples. The element in box 2 got a score of <u>20</u> and the element in box 3 got a score of <u>80</u> .

Output from the MLLM:

```
{ "analysis": "The element in Box 1, labeled 'Layers', appears to be a panel heading. Furthermore,
    based on the provided examples, it seems unlikely to open a previously unexplored or
    mysterious section. It might be less interactive compared to elements such as the 'Edit'
    dropdown.",
 "score": 15}
```

Figure 10: Example of the full MLLM input and output when predicting $\hat{Q}(a_x^t)$ through in-context learning. The input includes two example actions $(a_{\text{eg1}}, a_{\text{eg2}})$ marked by bounding boxes 2 and 3, and the candidate action a_x^t marked by bounding box 1. The prompt if formed by a fixed template with the GUI environment name, state s^t, and $(Q(a_{\text{eg1}}), Q(a_{\text{eg2}}))$ that are all underlined. The output is the predicted Q-value $\hat{Q}(a_x^t)$ for a_x^t.

Screenshot



A11y information (truncated)

```
link 'My Wish List 10 items' [0.651, 0.011, 0.769, 0.031]
link 'Sign Out' [0.781, 0.011, 0.828, 0.031]
StaticText 'Welcome, Emma Lopez!' [0.853, 0.011, 0.983, 0.031]
image 'one stop market logo' [0.017, 0.076, 0.185, 0.105]
link '\\ue611 My Cart 31 31items' [0.931, 0.076, 0.983, 0.111]
StaticText 'Search' [0.708, 0.110, 0.745, 0.130]
combobox '\\ue615 Search' autocomplete: both hasPopup: listbox
required: False expanded:... [0.710, 0.076, 0.920, 0.110]
link 'Advanced Search' [0.710, 0.116, 0.803, 0.136]
button 'Search' disabled: True [0.893, 0.076, 0.911, 0.110]
tablist '' multiselectable: False orientation: horizontal
[0.000, 0.137, 1.000, 0.251]
menuitem '\\ue622 Beauty & Personal Care' hasPopup: menu
[0.007, 0.137, 0.171, 0.194]
menuitem '\\ue622 Sports & Outdoors' hasPopup: menu [0.179,
0.137, 0.314, 0.194]
(More)
```

Query:

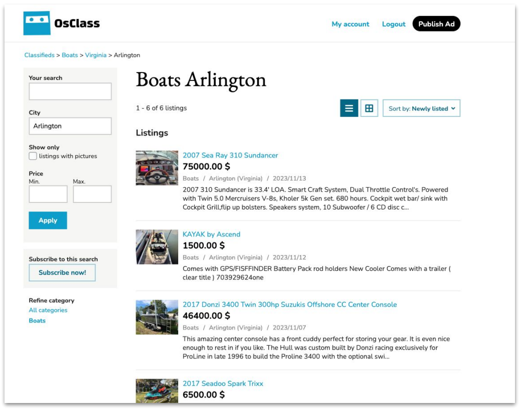
```
Add the Dairy Free CocoWhip Light to 'My
Wish List'
```

Ground truth grounding target:

```
[0.898, 0.867, 0.927, 0.904]
```

Figure 11: Example of NovelScreenSpot data from the Shopping environment.

Screenshot



A11y information (truncated)

```
link 'My account' [0.627, 0.029, 0.702, 0.067]
link 'Logout' [0.704, 0.029, 0.757, 0.067]
link 'Publish Ad' [0.759, 0.029, 0.832, 0.067]
link 'Classifieds' [0.170, 0.116, 0.216, 0.137]
StaticText '> ' [0.218, 0.116, 0.227, 0.137]
link 'Boats' [0.227, 0.116, 0.251, 0.137]
link 'Virginia' [0.262, 0.116, 0.295, 0.137]
StaticText 'Arlington' [0.305, 0.116, 0.346, 0.137]
group '' [0.177, 0.175, 0.302, 0.240]
heading 'Your search' [0.177, 0.175, 0.302, 0.193]
textbox '' required: False [0.177, 0.196, 0.302, 0.240]
group '' [0.177, 0.262, 0.302, 0.327]
heading 'City' [0.177, 0.262, 0.302, 0.280]
textbox '' required: False [0.177, 0.283, 0.302, 0.327]
StaticText 'Arlington' [0.183, 0.294, 0.226, 0.316]
group '' [0.177, 0.349, 0.302, 0.393]
heading 'Show only' [0.177, 0.349, 0.302, 0.367]
checkbox 'listings with pictures' checked: false [0.177,
0.370, 0.189, 0.390]
group '' [0.177, 0.415, 0.302, 0.498]
heading 'Price' [0.177, 0.415, 0.302, 0.433]
(More)
```

Query:

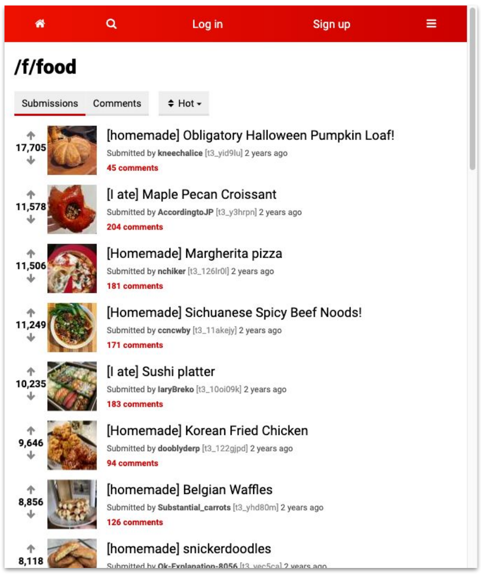
```
Check my personal listings.
```

Ground truth grounding target:

```
[0.627, 0.028, 0.701, 0.066]
```

Figure 12: Example of NovelScreenSpot data from the Classifieds environment.

Screenshot


Log in
Sign up
/f/food
Submissions
Comments
Hot
17,705
[homemade] Obligatory Halloween Pumpkin Loaf!
45 comments
11,578
[I ate] Maple Pecan Croissant
204 comments
11,506
[Homemade] Margherita pizza
181 comments
11,249
[Homemade] Sichuanese Spicy Beef Noods!
171 comments
10,235
[I ate] Sushi platter
183 comments
9,646
[Homemade] Korean Fried Chicken
94 comments
8,856
[homemade] Belgian Waffles
126 comments
8,118
[homemade] snickerdoodles

A11y information (truncated)

```
link 'Home' [0.000, 0.000, 0.151, 0.065]
image '' [0.064, 0.022, 0.087, 0.042]
button 'Places' hasPopup: menu expanded: False [0.827, 0.000,
0.978, 0.065]
image '' [0.890, 0.022, 0.914, 0.042]
button 'Search' hasPopup: menu expanded: False [0.151, 0.000,
0.302, 0.065]
image '' [0.215, 0.022, 0.238, 0.042]
link 'Log in' [0.302, 0.000, 0.557, 0.065]
StaticText 'Log in' [0.397, 0.021, 0.462, 0.044]
link 'Sign up' [0.557, 0.000, 0.827, 0.065]
StaticText 'Sign up' [0.653, 0.021, 0.731, 0.044]
heading '/f/food' [0.021, 0.083, 0.957, 0.135]
StaticText '/f/' [0.021, 0.088, 0.067, 0.129]
StaticText 'food' [0.067, 0.088, 0.152, 0.129]
link 'Submissions' [0.021, 0.152, 0.172, 0.196]
StaticText 'Submissions' [0.036, 0.164, 0.156, 0.184]
link 'Comments' [0.172, 0.152, 0.305, 0.196]
StaticText 'Comments' [0.187, 0.164, 0.289, 0.184]
button 'Sort by: Hot' hasPopup: menu expanded: False [0.326,
0.152, 0.432, 0.196]
image '' [0.341, 0.165, 0.362, 0.182]
StaticText 'Hot' [0.367, 0.164, 0.401, 0.184]
article '' [0.021, 0.214, 0.957, 0.301]
Obligatory Halloween Pumpkin Loaf!' [0.216, 0.217, 0.825,
0.244]
Obligatory Halloween Pumpkin Loaf!' [0.216, 0.217, 0.825,
0.244]
Obligatory Halloween Pumpkin Loaf!' [0.216, 0.217, 0.825,
0.244]
StaticText 'Submitted by ' [0.216, 0.254, 0.324, 0.271]
link 'kneechalice' expanded: False [0.324, 0.254, 0.419, 0.271]
StaticText 'kneechalice' [0.324, 0.254, 0.419, 0.271]
StaticText ' ' [0.419, 0.254, 0.424, 0.271]
(More)
```

Query:

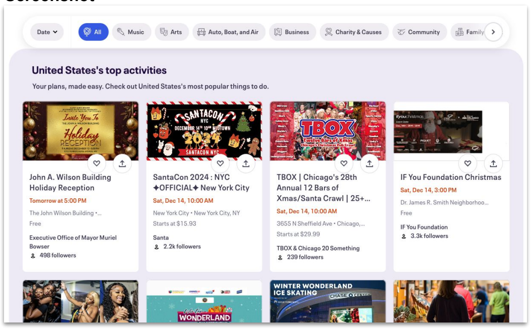
```
Go to 'Featured forums' section.
```

Ground truth grounding target:

```
[0.063, 0.021, 0.086, 0.041]
```

Figure 13: Example of NovelScreenSpot data from the Reddit environment.

Screenshot


Date
All
Music
Arts
Auto, Boat, and Air
Business
Charity & Causes
Community
Family
United States's top activities
Your plans, made easy. Check out United States's most popular things to do.
John A. Wilson Building Holiday Reception
SantaCon 2024 : NYC ✦OFFICIAL✦ New York City
TBOX | Chicago's 28th Annual 12 Bars of Xmas/Santa Crawl | 25+...
IF You Foundation Christmas

A11y information (truncated)

```
button 'Date' hasPopup: menu pressed: false expanded:
False controls: dateFilterDropdown [0.063, 0.056, 0.125,
0.111]
StaticText 'Date' [0.076, 0.070, 0.100, 0.096]
button 'All' pressed: true [0.148, 0.050, 0.209, 0.117]
StaticText 'All' [0.180, 0.070, 0.193, 0.096]
button 'Music' pressed: false [0.209, 0.050, 0.287, 0.117]
StaticText 'Music' [0.241, 0.070, 0.271, 0.096]
button 'Arts' pressed: false [0.287, 0.050, 0.356, 0.117]
StaticText 'Arts' [0.319, 0.070, 0.340, 0.096]
button 'Auto, Boat, and Air' pressed: false [0.356, 0.050,
0.496, 0.117]
StaticText 'Auto, Boat, and Air' [0.388, 0.070, 0.480,
0.096]
button 'Business' pressed: false [0.496, 0.050, 0.589,
0.117]
(More)
```

Query:

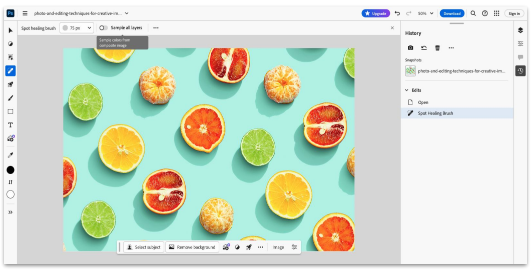
```
Click the share icon of 'IF You Foundation
Christmas' event card.
```

Ground truth grounding target:

```
[0.893, 0.470, 0.924, 0.525]
```

Figure 14: Example of NovelScreenSpot data from the Eventbrite environment.

Screenshot



A11y information (truncated)

```
generic '' describedby: sp-overlay-helper-571104f6
[0.005, 0.011, 0.020, 0.041]
button 'Main menu' hasPopup: menu expanded: False
describedby: option-picker [0.030, 0.005, 0.050, 0.047]
button
'photo-and-editing-techniques-for-creative-images'
hasPopup: menu expanded: False describedby:
option-picker [0.053, 0.005, 0.243, 0.047]
button 'Upgrade' [0.684, 0.011, 0.738, 0.042]
button 'Undo' [0.742, 0.005, 0.762, 0.047]
button 'Redo' disabled: True [0.764, 0.005, 0.784,
0.047]
button 'Zoom level' hasPopup: menu expanded: False
describedby: option-picker [0.786, 0.005, 0.827, 0.047]
(More)
```

Query:

```
Find the crop tool.
```

Ground truth grounding target:

```
[0.002, 0.070, 0.021, 0.112]
```

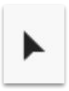


Figure 15: Example of NovelScreenSpot data from the Photoshop-web environment.