CGL: Advancing Continual GUI Learning via Reinforcement Fine-Tuning

Supplementary Material

1. Theoretical Analysis of Entropy Dynamics

To provide a rigorous theoretical grounding for our Entropy-Regulated Tuning strategy, we analyze the first-order dynamics of policy entropy $\mathcal{H}(\pi_\theta)$ under the joint influence of GRPO and SFT. Let $z_{s,a}$ denote the logits such that the policy is defined as $\pi_\theta(a|s) = \text{softmax}(z_{s,a}) = \frac{\exp(z_{s,a})}{\sum_{a'}\exp(z_{s,a'})}$.

1.1. Entropy Sensitivity and Separability

We first establish the relationship between logit updates and entropy variations.

Lemma 1 (Entropy-Covariance Relationship)**.** *The first-order approximation of the change in policy entropy $\Delta\mathcal{H}$ induced by a logit update vector $\Delta\mathbf{z}_s$ is determined by the negative covariance between the log-probabilities and the update values:*

$$\begin{aligned}\Delta\mathcal{H} &= \mathcal{H}(\pi_\theta^{k+1}|s) - \mathcal{H}(\pi_\theta^{k}|s) \\ &= -\text{Cov}_{a\sim\pi_\theta^k}\left(\log\pi_\theta^k(a|s), \Delta z_{s,a}\right) + o(\eta)\end{aligned} \tag{1}$$

where η is the learning rate.

Proof. The gradient of entropy with respect to logits is $\nabla_{z_{s,a}}\mathcal{H} = -\pi_\theta(a|s)(\log\pi_\theta(a|s) - \mathcal{H})$. The first-order Taylor expansion gives $\Delta\mathcal{H} \approx \sum_a \nabla_{z_{s,a}}\mathcal{H}\cdot\Delta z_{s,a}$. Substituting the gradient, we get $\Delta\mathcal{H} \approx -\mathbb{E}_a[\log\pi_\theta(a)\Delta z_{s,a}] + \mathcal{H}\mathbb{E}_a[\Delta z_{s,a}]$. This simplifies to the negative covariance definition $-\mathbb{E}[(\log\pi - \mathbb{E}[\log\pi])(\Delta z - \mathbb{E}[\Delta z])]$, assuming centered updates or simplified via the softmax invariance. □

Under our hybrid objective $\mathcal{L} = \mathcal{L}_{GRPO} + \lambda\mathcal{L}_{SFT}$, the total update decomposes linearly: $\Delta z_{total} = \Delta z^{GRPO} + \lambda\Delta z^{SFT}$. Due to the linearity of the covariance operator, the entropy dynamics are separable:

$$\Delta\mathcal{H}_{total} \approx \Delta\mathcal{H}_{GRPO} + \lambda\Delta\mathcal{H}_{SFT} \tag{2}$$

This property allows us to analyze the impact of GRPO and SFT independently.

1.2. Entropy Decay in GRPO: The Matthew Effect

The GRPO algorithm optimizes the expected relative advantage. We show that this naturally leads to entropy reduction.

Lemma 2 (GRPO Update Dynamics)**.** *The logit update under GRPO, denoted as $\Delta z_{s,a}^{GRPO}$, is proportional to the probability-weighted advantage:*

$$\Delta z_{s,a}^{GRPO} = \eta\cdot\pi_\theta(a|s)\cdot A(s,a) \tag{3}$$

assuming the mean advantage baseline is handled implicitly.

Proof. The gradient of the GRPO objective $J = \mathbb{E}_{a\sim\pi}[A(s,a)]$ with respect to logit $z_{s,a}$ is:

$$\begin{aligned}\frac{\partial J}{\partial z_{s,a}} &= \mathbb{E}_{a'\sim\pi}\left[\frac{\partial\log\pi(a'|s)}{\partial z_{s,a}}A(s,a')\right] \\ &= \sum_{a'}\pi(a'|s)(\mathbb{I}[a=a'] - \pi(a|s))A(s,a') \\ &= \pi(a|s)\left(A(s,a) - \sum_{a'}\pi(a'|s)A(s,a')\right)\end{aligned} \tag{4}$$

In GRPO, advantages are normalized group-wise such that $\sum\pi A \approx 0$. Thus, the update simplifies to $\Delta z_{s,a} = \eta\frac{\partial J}{\partial z_{s,a}} \approx \eta\pi(a|s)A(s,a)$. □

Analysis (The Matthew Effect): Eq. (3) reveals a self-reinforcing mechanism. Actions with initially high probability $\pi(a|s)$ and positive advantage $A(s,a) > 0$ receive the largest positive updates.

- High $\pi(a|s)$ implies high $\log\pi(a|s)$.
- High $\pi(a|s)$ leads to large positive $\Delta z_{s,a}$.

This creates a **positive correlation** between the distribution and the update: $\text{Cov}(\log\pi, \Delta z^{GRPO}) > 0$. According to Lemma 1, this results in $\Delta\mathcal{H}_{GRPO} < 0$. This mathematically explains the "Stage 3" behavior where the model converges sharply to a low-entropy state.

1.3. Entropy Injection via Error-Driven SFT

When the model exhibits pathological bias (i.e., convergence to a wrong action), we activate SFT on the ground truth a^*.

1.4. Entropy Injection via Error-Driven SFT

When the model suffers from pathological bias (Stage 1), we activate SFT on the ground-truth action a^*.

Lemma 3 (SFT Entropy Injection)**.** *Let the SFT update follow the gradient of the target probability likelihood. The logit update Δz^{SFT} satisfies the zero-sum property and induces entropy injection by creating a negative covariance with the current distribution.*

Proof. Consider the optimization of the probability mass $\pi_\theta(a^*|s)$ directly. Utilizing the derivative property of the Softmax function, $\frac{\partial\pi_i}{\partial z_j} = \pi_i(\delta_{ij} - \pi_j)$, the update for logit

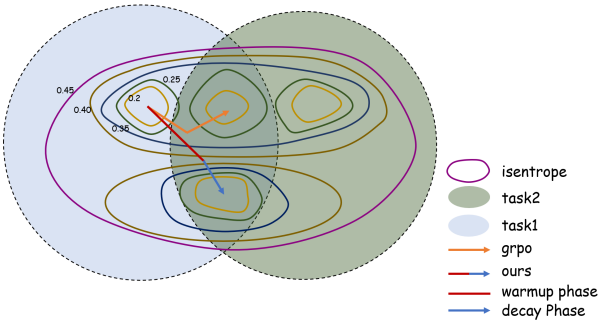

Figure 1. High-accuracy region comparison of GRPO and our CGL in continual GUI learning.

$z_{s,a}$ is derived as:

$$\begin{aligned}\Delta z_{s,a}^{SFT} &= \eta \cdot \frac{\partial \pi_\theta(a^*|s)}{\partial z_{s,a}} \\ &= \eta \cdot \pi_\theta(a^*|s) \cdot (\mathbb{I}[a = a^*] - \pi_\theta(a|s))\end{aligned} \tag{5}$$

We verify the zero-sum property of this update:

$$\sum_a \Delta z_{s,a}^{SFT} = \eta \pi_\theta(a^*|s) \left(\sum_a \mathbb{I}[a = a^*] - \sum_a \pi_\theta(a|s) \right) = 0 \tag{6}$$

This confirms the update purely redistributes probability mass. We then analyze the correlation structure in the error state ($\pi_\theta(a^*|s) \to 0$):

1. **For the target $a = a^*$:** The current log-probability $\log \pi_\theta$ is minimal (negative), while the update component $(\mathbb{I} - \pi_\theta) > 0$ is positive.
2. **For erroneous actions $a \neq a^*$:** The current log-probability $\log \pi_\theta$ is high (confident error), while the update component $(0 - \pi_\theta) < 0$ is negative.

Despite the scaling factor $\pi_\theta(a^*|s)$, the direction of Δz is strictly opposite to the bias of π_θ. Thus, $\mathrm{Cov}(\log \pi_\theta, \Delta z^{SFT}) < 0$. □

Analysis: According to Lemma 1, the negative covariance implies $\Delta \mathcal{H}_{SFT} > 0$. By increasing the weight λ in **Stage 2**, the framework amplifies this corrective update. Even if the magnitude is initially dampened by the small $\pi_\theta(a^*|s)$, the directional alignment forces the "heating up" of the distribution, breaking the local minima of the initial error.

Mechanism Advantage: The isentrope diagram (Fig. 1) illustrates that GRPO is confined to existing isentropic regions, while our entropy-regulated tuning (driven by dynamic λ) enables crossing isentropes—jumping from low-entropy local optima to more favorable regions, thus avoiding suboptimal policy stagnation.

2. Reward Function For Gui Agent

To guide effective policy learning, we categorize UI actions into three classes with tailored reward rules.

- **Navigation and State Actions**: Include `Home()`, `BACK()`, `WAIT()`, `Finish()`.Reward: +1 for exact action type match with ground truth (GT); 0 otherwise.
- **Parameterized Actions**: Include `INPUT_TEXT(str)`, `SCROLL(direction)`,`OPEN_APP(app_name)`.Reward: +1 for type match (Stage 1),+ 1 for exact parameter match (Stage 2), total maximum +2.
- **Spatial Interaction Actions**: Include `CLICK([x,y])`, `LONG_PRESS([x,y])`.Reward: +1 for type match (Stage 1) + 1 if predicted coordinates fall within GT element's bounding box (Stage 2), total maximum +2.

3. AndroidControl-CL Benchmark

Continual learning for GUI agents is essential in the face of rapidly emerging mobile applications and frequent software updates, yet existing datasets lack dedicated benchmarks tailored to dynamic GUI environments. To address this gap, we propose the **AndroidControl-CL** benchmark, an extension of AndroidControl [**?**] with additional annotations and structured task splits. This benchmark is specifically designed for the systematic evaluation of Continual GUI Learning (CGL).

Explicit, Multi-Source App Identity Annotation. The original dataset only includes sparse app identifiers inferred from occasional `open_app` actions. AndroidControl-CL introduces a compact and robust attribution pipeline that integrates three complementary cues: (1) extracting explicit app identities from `open_app` parameters; (2) identifying direct app references in natural-language instructions; and (3) performing screenshot-based app recognition when textual cues are insufficient. This yields clear, instance-level app identity labels.

Functional Super-Class Based Task Splits. Building upon the per-trajectory app annotations, we structure AndroidControl-CL into a sequence of learning tasks based on app-level semantic categories. Specifically, we group apps into seven functional super-classes that reflect common mobile application domains: Shopping (SP), Productivity and Office (PO), Communication (CO), Travel and Transportation (TT), System Tools (ST), Education and Science (ES), and Life and Entertainment (LE). This design is motivated by the observation that apps within the same super-class tend to exhibit similar task logic, UI layouts, and interaction patterns, whereas those from different super-classes often present substantial distributional and functional gaps. In each task, we split the training and test sets

Table 1. App Category Statistics for Training Dataset

Category	Number of Apps	Number of Episodes	App Examples
Shopping	15	1140	amazon, ebay, alibaba.com
Productivity	15	1240	Drive, WPS Office, Word
Communication	8	617	Gmail, Outlook, Skype
Travel	10	686	Maps, booking.com, Expedia
Tools	13	771	Clock, Recorder, Settings
Education	11	559	Leafsnap, Arts & Culture, Duolingo
Entertainment	17	1030	YouTube, Vimeo, Pinterest

Table 2. **Evolution of Task-Specific Performance (LLaVA-OneVision-0.5b, Task Order 1).** Columns represent the sequence of training tasks (left to right); rows represent the evaluation performance on specific test sets after each training stage.

Training Stage →	Shopping	Productivity	Communication	Travel	Tools	Education	Entertainment
Test on:							
Shopping	77.63	75.63	75.58	76.58	76.12	76.49	75.55
Productivity	-	80.80	80.80	78.39	80.97	81.30	81.32
Communication	-	-	80.88	77.90	80.77	80.33	80.73
Travel	-	-	-	77.78	77.09	76.27	76.88
Tools	-	-	-	-	84.14	84.09	83.41
Education	-	-	-	-	-	69.11	68.90
Entertainment	-	-	-	-	-	-	78.13

Table 3. **Evolution of Task-Specific Performance (CGL on Qwen2.5-VL-3b, Task Order 1).** CGL demonstrates stability and positive transfer (e.g., Tools accuracy improves over time).

Training Stage →	Shopping	Productivity	Communication	Travel	Tools	Education	Entertainment
Test on:							
Shopping	79.77	80.67	81.33	81.77	80.53	81.72	80.32
Productivity	-	85.41	85.18	84.13	85.46	85.29	85.46
Communication	-	-	84.23	83.98	82.98	83.20	81.99
Travel	-	-	-	83.06	83.13	82.51	82.03
Tools	-	-	-	-	86.42	87.88	88.17
Education	-	-	-	-	-	75.05	75.81
Entertainment	-	-	-	-	-	-	82.53

Table 4. Distribution of AndroidControl-CL. Abbreviations: SP (Shopping), PO (Productivity and Office), CO (Communication), TT (Travel and Transportation), ST (System Tools), ES (Education and Science), LE (Life and Entertainment).

Categories	**SP**	**PO**	**CO**	**TT**	**ST**	**ES**	**LE**
Apps	15	15	8	10	13	11	17
Trajectories	1140	1240	617	686	771	559	1030

at the trajectory level with an 8:2 ratio. By organizing tasks along these super-class boundaries, we establish a realistic and challenging continual learning setting that evaluates an agent's ability to incrementally acquire and generalize knowledge across distinct app domains.

Balanced Task Distribution. After task partitioning, we observed data imbalance across two dimensions: (1) imbalance numbers of apps across tasks, and (2) imbalance numbers of trajectory across apps. To ensure a fair evaluation setup, we perform targeted data filtering to balance both the number of apps per task and the number of trajectories per app. The resulting balanced task distribution is shown in Tab. 4.

Fine-Grained Bounding Box Annotation for Click Actions. In the original AndroidControl dataset, click actions are annotated only as single-point coordinates. This fails to capture the realistic nature of GUI interactions,

where a user typically targets a UI element (e.g., a button or icon) that occupies a spatial region rather than an infinitesimal point. To better reflect this, we enhance the annotation by assigning each click action a bounding box that delineates the corresponding interactive UI element. This is achieved through a designed three-stage pipeline: (1) automatic UI element parsing using OmniParser [**?**] to extract candidate bounding boxes from screenshots; (2) matching the original annotated click coordinate to the nearest parsed UI element to obtain an initial bounding box assignment; and (3) manual verification and correction to resolve alignment errors or ambiguous cases. The resulting region-level annotations provide more semantically meaningful supervision, enabling GUI agents to learn grounded, element-aware action policies with higher fidelity.

4. Additional Experimental Results

This section provides further empirical evidence supporting the effectiveness of the CGL framework compared to baselines (GRPO, SFT). We analyze performance across models of varying parameter scales and contrast sequential learning with multi-task joint training. Unless otherwise stated, Task Order 1 is adopted as the evaluation protocol.

4.1. CGL Performance Across Models of Varying Scales

To validate the scalability and generality of the CGL framework, we conducted experiments on two multimodal models with distinct parameter sizes: the lightweight **LLaVA-OneVision-0.5b** and the large-scale **Qwen2.5-VL-3b**.

Performance on Lightweight Model (0.5B): Tab. 2 presents the step-wise accuracy evolution for LLaVA-OneVision-0.5b. Despite the limited model capacity, CGL maintains remarkable stability. For instance, after the initial training on the *Shopping* task, the model preserves high performance on subsequent evaluations even as it adapts to new domains (*e.g.* retaining $\sim$76% on Shopping after learning Education), demonstrating effective mitigation of forgetting in resource-constrained settings.

Performance on Large-scale Model (3B): As shown in Tab. 3, CGL demonstrates superior capability on Qwen2.5-VL-3b, achieving not only retention but also **positive backward transfer**. Key tasks such as *Shopping*, *Tools*, and *Productivity* exhibit performance maintenance or improvement as new tasks are introduced. Specifically, the accuracy on *Tools* increases from 86.42% to 88.17%, and *Productivity* improves marginally from 85.29% to 85.46%. This suggests that the synergy between SFT and GRPO in CGL enables the consolidation of generalizable GUI interaction logic.

Quantitatively, CGL achieves a near-zero Forgetting Measure (FM = -0.02) under Task Order 1. Notably, under Task Order 2 (Tab. 6), CGL achieves a positive FM (+0.13). This indicates that the learning of subsequent tasks actively reinforces the representations of previously learned tasks, a phenomenon rarely observed in standard continual learning baselines.

4.2. Baseline Analysis: GRPO and SFT

To contextualize the performance of CGL, we analyze the behavior of baseline methods on Qwen2.5-VL-3b:

- **GRPO (Tab. 7):** While GRPO demonstrates strong stability and resistance to catastrophic forgetting, it lacks the rapid adaptation capabilities observed in CGL, serving primarily as a stability anchor.
- **SFT (Tab. 8):** Standard Supervised Fine-Tuning exhibits severe plasticity-stability trade-offs. For instance, after adapting to the *Entertainment* task, performance on early tasks such as *Shopping* drops significantly (79.77% $\rightarrow$ 70.54%), confirming that SFT prioritizes new task acquisition at the expense of catastrophic forgetting.

4.3. Comparison with Multi-Task Joint Training

We further compare our sequential learning approach against the theoretical upper bound of Multi-Task Joint Training (Tab. 9). Joint training assumes simultaneous access to all datasets, which is typically impractical for dynamic, real-world GUI agents.

Remarkably, CGL significantly narrows the gap between sequential learning and joint training. As illustrated in Tab. 9:

- The average trajectory accuracy of CGL across three different task orders is **82.33%** (Order 1), **82.61%** (Order 2), and **82.40%** (Order 3).
- These results are highly competitive with the **GRPO-Joint-Training** baseline, which achieves **82.41%**. Notably, our method under Task Order 2 even slightly outperforms the GRPO joint training baseline.
- While **SFT-Joint-Training** reaches the highest absolute accuracy (83.48%), it requires full data accessibility. CGL achieves comparable performance to the RL-based joint training upper bound while strictly adhering to the sequential learning constraint, validating the efficacy of our Entropy-Regulated Tuning and Gradient Surgery mechanisms in maintaining policy optimality over time.

Table 5. Qwen2.5VL-7B Results.

Method	Step (%) ↑	Traj. (%) ↑	FM ↑
SFT	80.50	35.12	-4.78
GRPO	83.17	40.39	-0.36
Ours	**83.85**	**42.82**	**-0.03**

Table 6. **Evolution of Task-Specific Performance (CGL on Qwen2.5-VL-3b, Task Order 2).** The distinct task order further validates the robustness of CGL.

Training Stage →	Shopping	Entertainment	Education	Tools	Travel	Communication	Productivity
Test on:							
Shopping	79.77	79.96	80.53	80.72	80.58	80.25	79.63
Entertainment	-	81.55	81.45	81.05	80.71	81.45	82.08
Education	-	-	76.67	75.81	76.03	75.27	75.81
Tools	-	-	-	86.52	86.23	86.61	87.29
Travel	-	-	-	-	81.96	82.10	82.85
Communication	-	-	-	-	-	84.75	84.42
Productivity	-	-	-	-	-	-	86.18

Table 7. **Baseline Performance: GRPO (Qwen2.5-VL-3b, Task Order 1).** GRPO shows high stability but lower plasticity compared to CGL.

Training Stage →	Shopping	Productivity	Communication	Travel	Tools	Education	Entertainment
Test on:							
Shopping	79.77	81.39	80.30	80.34	79.72	80.34	79.63
Productivity	-	85.22	84.18	83.63	84.13	84.46	84.74
Communication	-	-	84.20	82.65	83.20	82.87	83.43
Travel	-	-	-	81.98	80.59	80.79	81.34
Tools	-	-	-	-	86.71	85.35	85.74
Education	-	-	-	-	-	75.05	74.08
Entertainment	-	-	-	-	-	-	81.73

Table 8. **Baseline Performance: SFT (Qwen2.5-VL-3b, Task Order 1).** SFT suffers from significant forgetting (e.g., Shopping drops from 79.77% to 70.54%).

Training Stage →	Shopping	Productivity	Communication	Travel	Tools	Education	Entertainment
Test on:							
Shopping	79.77	75.62	73.59	73.34	71.57	70.89	70.54
Productivity	-	86.49	82.97	81.92	80.89	80.12	79.52
Communication	-	-	85.30	81.80	80.23	79.67	78.90
Travel	-	-	-	82.06	79.78	78.45	76.41
Tools	-	-	-	-	87.81	85.01	83.24
Education	-	-	-	-	-	74.89	69.24
Entertainment	-	-	-	-	-	-	80.42

Table 9. **Comparison with Multi-Task Joint Training (Qwen2.5-VL-3b).** Joint training represents the theoretical upper bound where all data is available simultaneously. CGL (Ours) achieves performance comparable to GRPO-Joint-Training across all task orders.

Training Method	Shopping	Productivity	Communication	Travel	Tools	Education	Entertainment	Avg. Step Acc.
Ours-Order1	80.32	85.46	81.99	82.03	**88.17**	<u>75.81</u>	<u>82.53</u>	82.33
Ours-Order2	79.63	<u>86.18</u>	84.42	**82.85**	87.29	<u>75.81</u>	82.08	<u>82.61</u>
Ours-Order3	80.39	86.17	<u>84.91</u>	81.81	87.87	74.16	81.50	82.40
SFT-Joint-Training	**81.73**	**86.74**	**85.75**	82.24	**88.17**	**77.00**	**82.76**	**83.48**
GRPO-Joint-Training	<u>80.58</u>	85.97	84.53	<u>82.65</u>	87.49	74.73	80.94	82.41

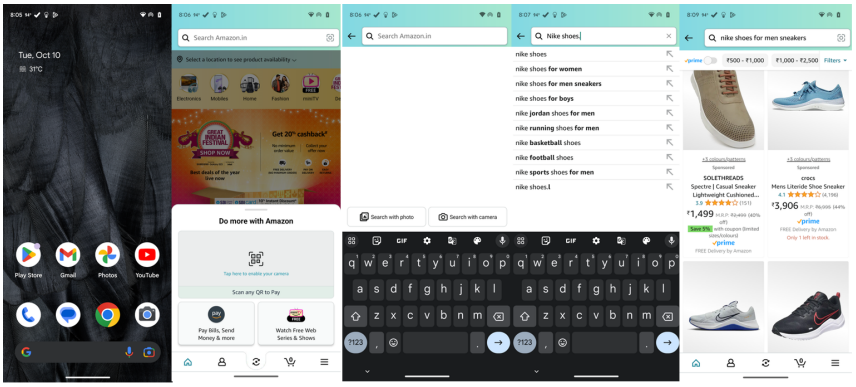

Figure 2. **Trajectory Visualization: Search for New Nike Shoes on Amazon.** The agent executes the task through the following action sequence: (1) Launch the application via `Open(Amazon)`; (2) Focus on the search field via `Click(0.507, 0.089)`; (3) Input the product query via `Input_text(''Nike shoes'')`; (4) Select the target suggestion via `Click(0.444, 0.210)`; (5) Browse the product list via `Scroll(down)`; (6) Terminate the episode via `Finish()`. Coordinates indicate the relative (x, y) position on the screen interface.

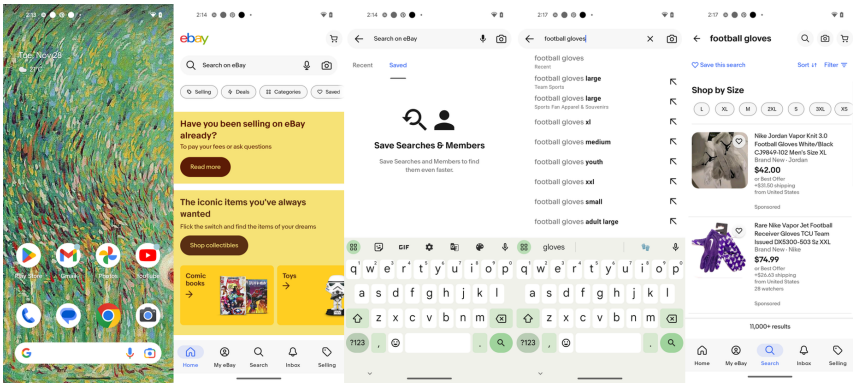

Figure 3. **Trajectory Visualization: Product Search on eBay.** The agent locates specific sporting goods via the following sequence: (1) Launch the application via `Open(eBay)`; (2) Activate the search interface via `Click(0.382, 0.162)`; (3) Input the product category via `Input_text(''football gloves'')`; (4) Select the top suggestion via `Click(0.490, 0.152)`; (5) Terminate the task via `Finish()`.

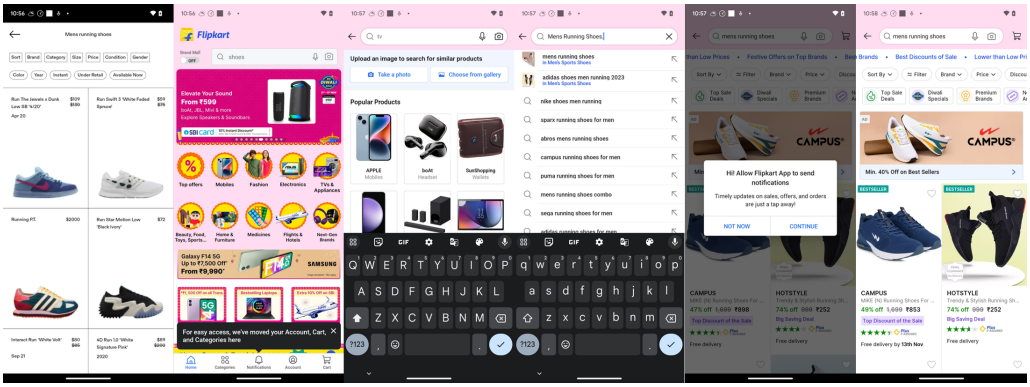

Figure 4. **Trajectory Visualization: Search for Mens Running Shoes on Flipkart.** The agent successfully navigates the interface through the following action sequence: (1) Launch the application via `Open(Flipkart)`; (2) Activate the search field via `Click(0.562, 0.139)`; (3) Enter the search query via `Input_text(''Mens Running Shoes'')`; (4) Select the target product from the results via `Click(0.318, 0.154)`; (5) Complete the task via `Finish()`. Coordinates (x, y) represent the normalized relative position on the screen.

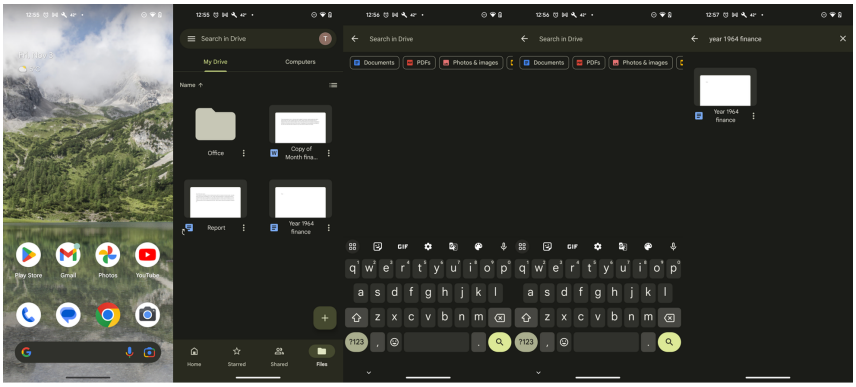

Figure 5. **Trajectory Visualization: File Retrieval on Google Drive.** The agent retrieves a specific historical document: (1) Launch the application via `Open(Drive)`; (2) Focus on the search bar via `Click(0.500, 0.091)`; (3) Enter the file query via `Input_text(''Year 1964 finance'')`; (4) Access the target file from results via `Click(0.495, 0.159)`; (5) Terminate the task via `Finish()`.

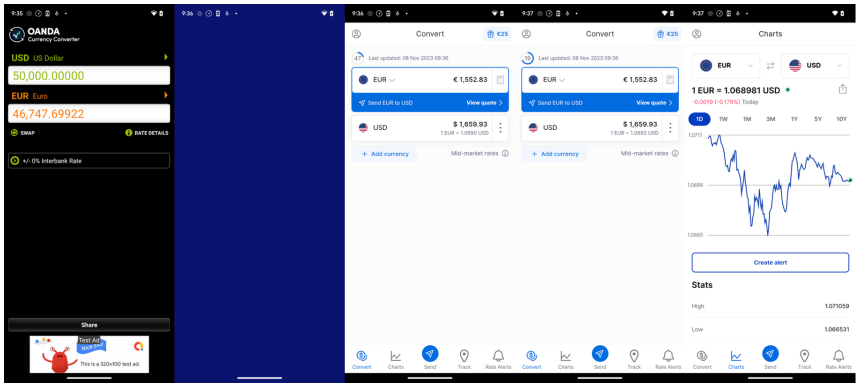

Figure 6. **Trajectory Visualization: Currency Exchange Tracking on Xe.** The agent navigates to the financial charting interface: (1) Launch the application via `Open(Xe)`; (2) Synchronize with the interface via `Wait()`; (3) Execute a secondary `Wait()` for data loading; (4) Navigate to the charts section via `Click(0.300, 0.937)`; (5) Terminate the task via `Finish()`.

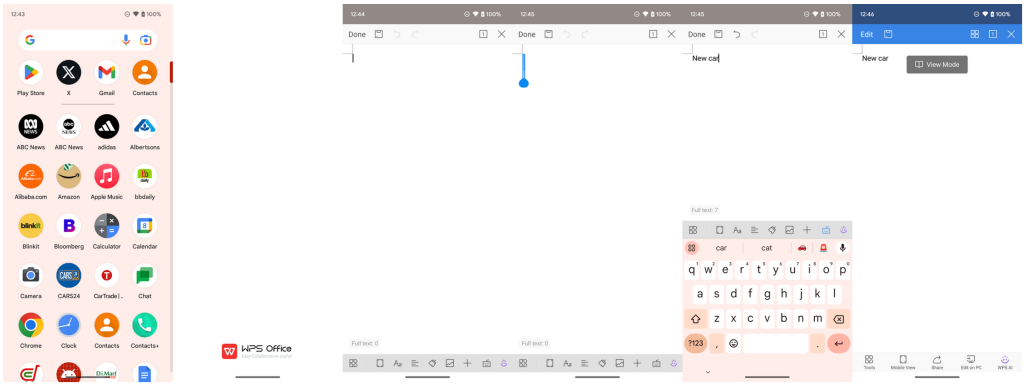

Figure 7. **Trajectory Visualization: Document Creation on WPS Office.** The agent initializes a new document: (1) Launch the application via `Open(WPS Office)`; (2) `Wait()` for initialization; (3) Input the document title via `Input_text(''New car'')`; (4) `Wait()` for processing; (5) Confirm the creation via `Click(0.079, 0.080)`; (6) Terminate the task via `Finish()`.

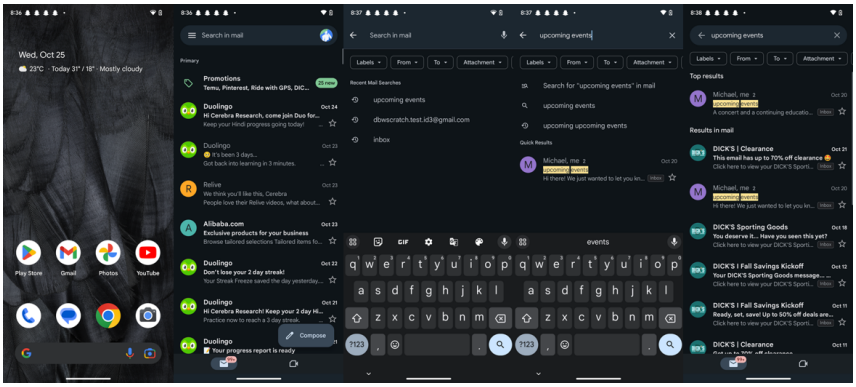

Figure 8. **Trajectory Visualization: Email Search on Gmail.** The agent filters inbox contents for specific events: (1) Launch the application via `Open(Gmail)`; (2) Focus on the search field via `Click(0.287, 0.084)`; (3) Enter the query via `Input_text(``upcoming events'')`; (4) Execute the search via `Click(0.920, 0.904)`; (5) Terminate the task via `Finish()`.

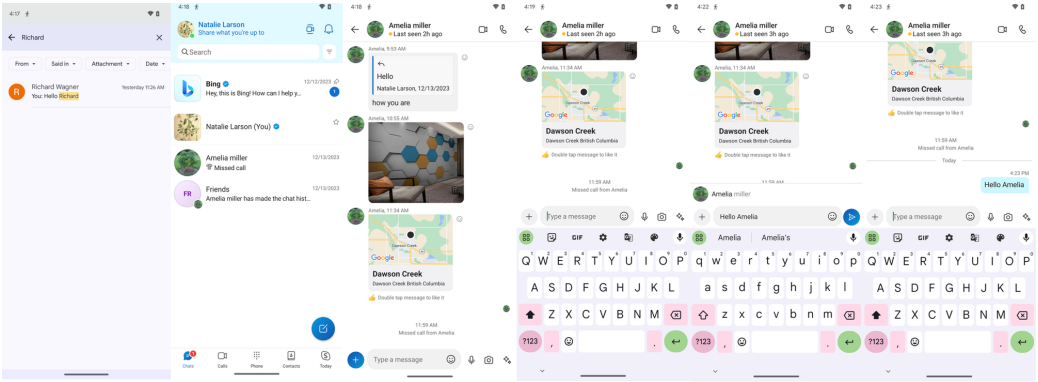

Figure 9. **Trajectory Visualization: Messaging on Skype.** The agent sends a greeting to a specific contact: (1) Launch the application via `Open(Skype)`; (2) Select the contact via `Click(0.500, 0.435)`; (3) Activate the text input area via `Click(0.378, 0.942)`; (4) Type the message via `Input_text(``Hello Amelia'')`; (5) Send the message via `Click(0.934, 0.574)`; (6) Terminate the task via `Finish()`.

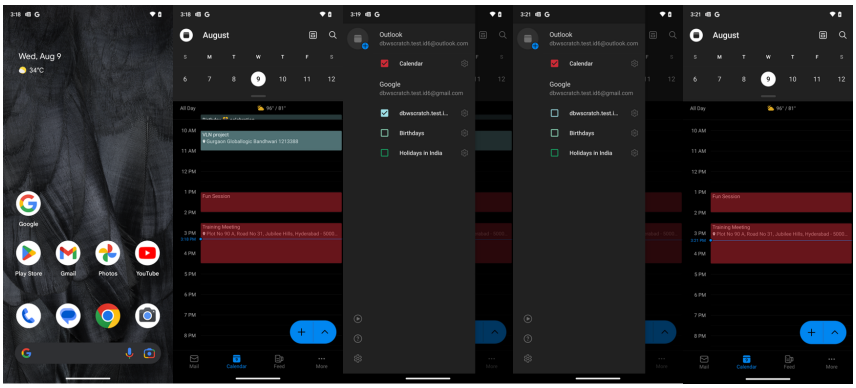

Figure 10. **Trajectory Visualization: Account Management on Outlook.** The agent navigates settings to manage accounts: (1) Launch the application via `Open(Outlook)`; (2) Open the side menu via `Click(0.078, 0.080)`; (3) Select the target account via `Click(0.244, 0.285)`; (4) Confirm the action via `Click(0.862, 0.387)`; (5) Terminate the task via `Finish()`.

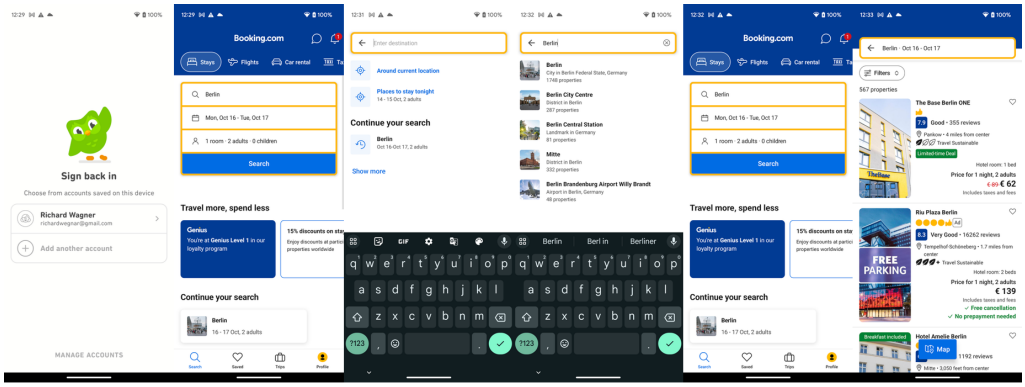

Figure 11. **Trajectory Visualization: Hotel Search on Booking.com.** The agent searches for accommodation in a specific location: (1) Launch the application via `Open(Booking.com)`; (2) Activate search via `Click(0.500, 0.239)`; (3) Input the destination via `Input_text(''Berlin'')`; (4) Select the suggestion via `Click(0.578, 0.181)`; (5) View results via `Click(0.500, 0.422)`; (6) Terminate the task via `Finish()`.

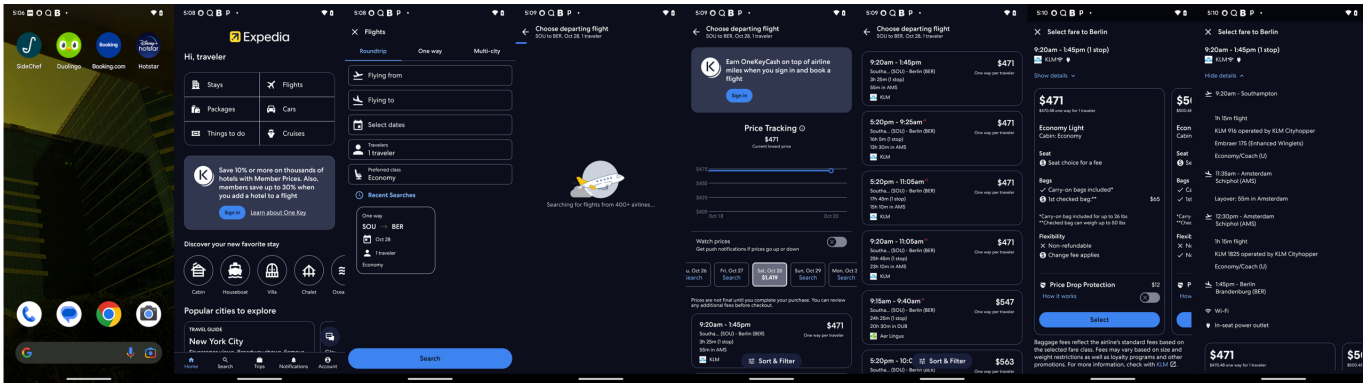

Figure 12. **Trajectory Visualization: Flight Details on Expedia.** The agent retrieves specific flight information: (1) Launch the application via `Open(Expedia)`; (2) Select flights via `Click(0.808, 0.203)`; (3) Access recent searches via `Click(0.309, 0.586)`; (4) `Wait()` for data retrieval; (5) Browse results via `Scroll(down)`; (6) Select the airline via `Click(0.500, 0.196)`; (7) View detailed itinerary via `Click(0.137, 0.188)`; (8) Terminate the task via `Finish()`.

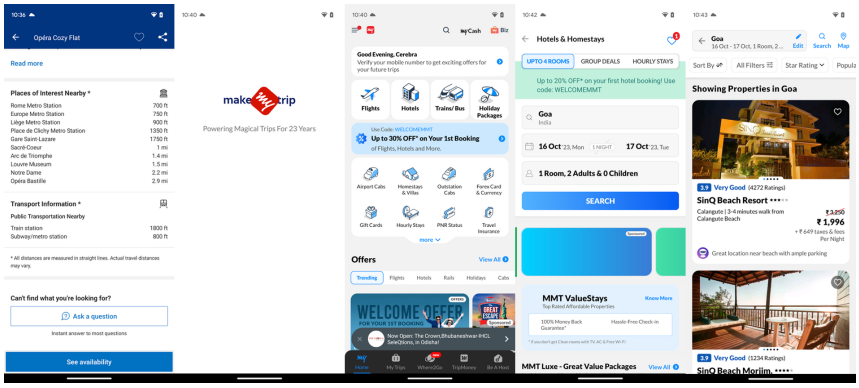

Figure 13. **Trajectory Visualization: Hotel Inquiry on MakeMyTrip.** The agent initiates a hotel search process: (1) Launch the application via `Open(MakeMyTrip)`; (2) `Wait()` for the interface to load; (3) Select the hotel category via `Click(0.383, 0.252)`; (4) Initiate the search via `Click(0.500, 0.518)`; (5) Terminate the task via `Finish()`.

Figure 14. **Trajectory Visualization: Alarm Adjustment on Clock.** The agent modifies an existing alarm setting: (1) Launch the application via `Open(Clock)`; (2) Select the target alarm via `Click(0.233, 0.226)`; (3) Adjust the hour setting via `Click(0.379, 0.660)`; (4) Confirm the modification via `Click(0.801, 0.760)`; (5) Terminate the task via `Finish()`.

Figure 15. **Trajectory Visualization: Forecast Retrieval on Weather.** The agent navigates to the extended forecast view: (1) Launch the application via `Open(Weather)`; (2) `Wait()` for data synchronization; (3) Expand the forecast view via `Click(0.905, 0.858)`; (4) Browse the weekly data via `Scroll(down)`; (5) Continue browsing via `Scroll(down)`; (6) Terminate the task via `Finish()`.

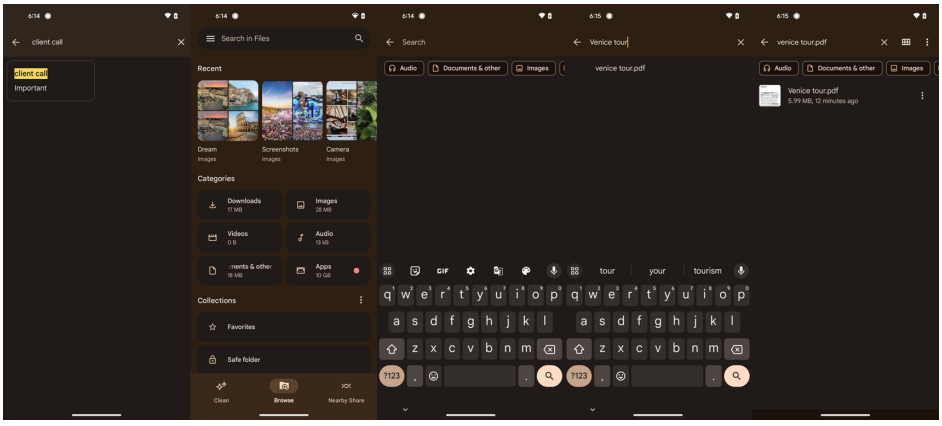

Figure 16. **Trajectory Visualization: Document Retrieval on Google Files.** The agent locates a specific file: (1) Launch the application via `Open(files)`; (2) Activate the search bar via `Click(0.902, 0.083)`; (3) Input the filename via `Input_text(''Venice tour'')`; (4) Open the file via `Click(0.524, 0.154)`; (5) Terminate the task via `Finish()`.

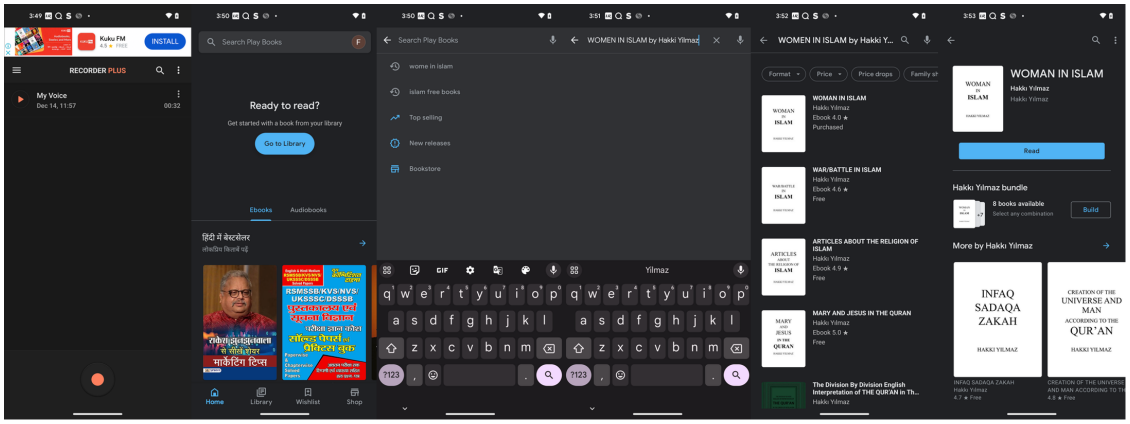

Figure 17. **Trajectory Visualization: Book Search on Google Play Books.** The agent searches for and selects a specific title: (1) Launch the application via `Open(google play books)`; (2) Focus on the search bar via `Click(0.324, 0.091)`; (3) Enter the book title via `Input_text(``WOMEN IN ISLAM by...'')`; (4) Execute search via `Click(0.920, 0.899)`; (5) Select the first result via `Click(0.500, 0.288)`; (6) Terminate the task via `Finish()`.

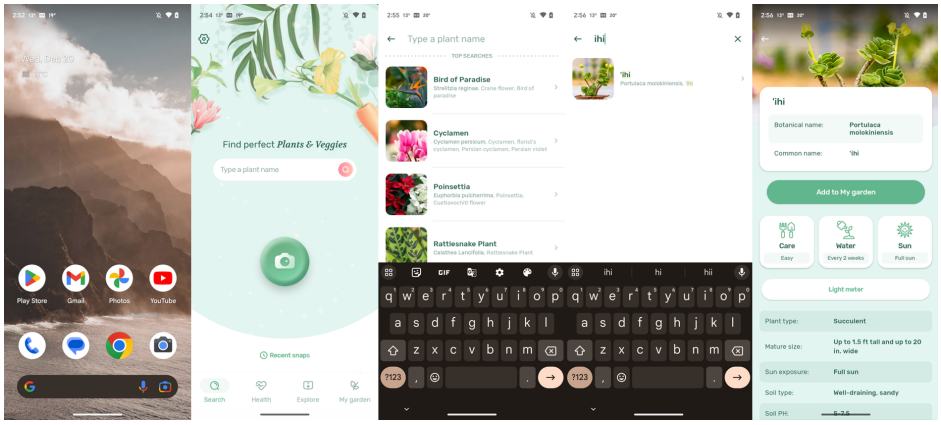

Figure 18. **Trajectory Visualization: Plant Identification on Blossom.** The agent searches for botanical information: (1) Launch the application via `Open(Blossom)`; (2) Activate the search bar via `Click(0.500, 0.398)`; (3) Input the plant name via `Input_text(``ihi'')`; (4) Select the entry via `Click(0.225, 0.177)`; (5) Terminate the task via `Finish()`.

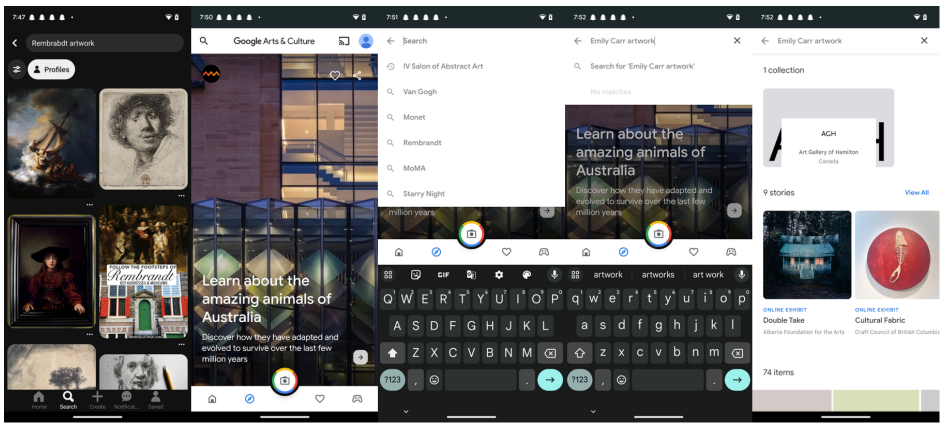

Figure 19. **Trajectory Visualization: Artwork Search on Arts & Culture.** The agent retrieves specific artist works: (1) Launch the application via `Open(Arts & Culture)`; (2) Click the search icon via `Click(0.067, 0.083)`; (3) Input the query via `Input_text(``Emily Carr artwork'')`; (4) Confirm the search via `Click(0.920, 0.904)`; (5) Terminate the task via `Finish()`.

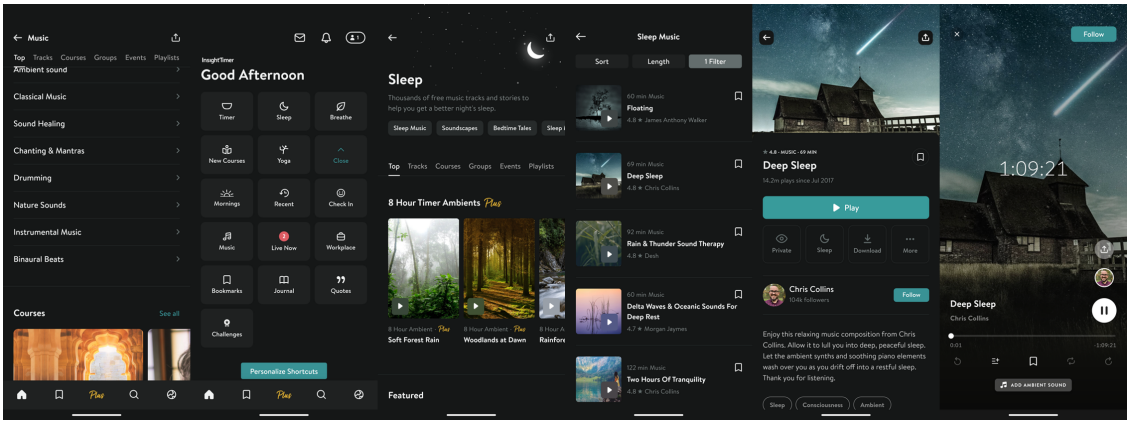

Figure 20. **Trajectory Visualization: Audio Playback on Insight Timer.** The agent navigates to and plays specific sleep content: (1) Navigate back via `Back()`; (2) Open the Sleep tab via `Click(0.499, 0.258)`; (3) Select the music category via `Click(0.172, 0.297)`; (4) Select the track "Deep sleep" via `Click(0.500, 0.412)`; (5) Activate playback via `Click(0.499, 0.489)`; (6) Terminate the task via `Finish()`.

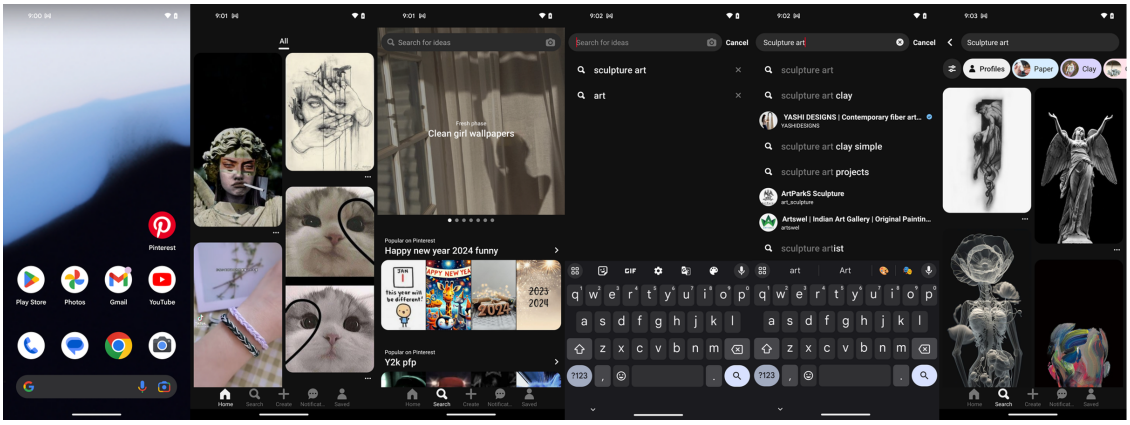

Figure 21. **Trajectory Visualization: Content Discovery on Pinterest.** The agent searches for a specific art category: (1) Launch the application via `Open(Pinterest)`; (2) Navigate to the search tab via `Click(0.344, 0.947)`; (3) Activate the search bar via `Click(0.494, 0.092)`; (4) Input the query via `Input_text(''Sculpture art'')`; (5) Execute search via `Click(0.920, 0.899)`; (6) Terminate the task via `Finish()`.

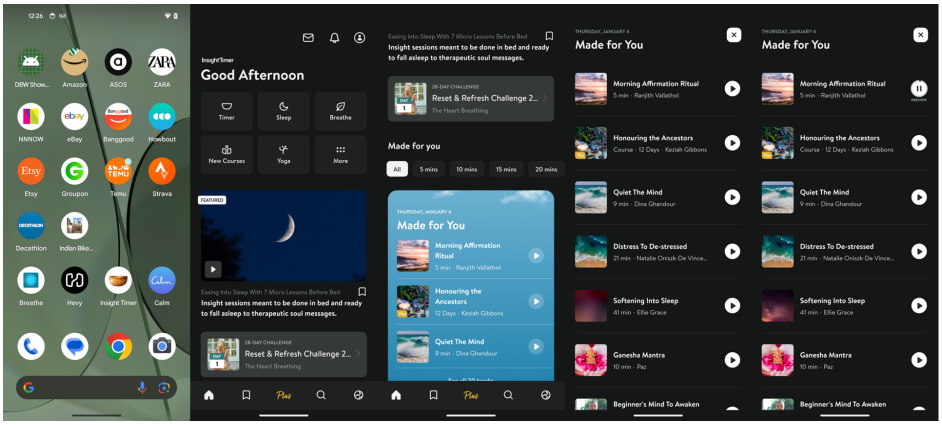

Figure 22. **Trajectory Visualization: Guided Meditation on Insight Timer.** The agent selects a morning ritual: (1) Launch the application via `Open(Insight Timer)`; (2) Browse content via `Scroll(down)`; (3) Select the "Morning Affirmation" podcast via `Click(0.499, 0.603)`; (4) Initiate playback via `Click(0.500, 0.203)`; (5) Terminate the task via `Finish()`.