arXiv:2606.06322v1 [cs.AI] 4 Jun 2026

DragOn: A Benchmark and Dataset for Drag-Based GUI Interactions

Nathan Bout [1] **Maxime Langevin** [1] **Ronan Riochet** [1]

Abstract

GUI agents – vision-based models that control desktops, web browsers, and mobile devices through graphical user interfaces – promise to automate a wide range of digital tasks. While million-scale datasets have enabled substantial progress on click-grounding, drag grounding (e.g. drag-and-drop, swipe, highlight) data remains an order of magnitude smaller and current models fall short on complex drag-based interactions. We introduce *DragOn*, a drag grounding benchmark and training dataset covering four domains: text highlighting, cell selection, element resizing and slider manipulation. The dataset comprises 286K training screenshots and 3.5M training tasks, plus a 2,000-example held-out evaluation suite. We evaluate proprietary (GPT, Claude) and open-weight (Qwen, Kimi, Holo) models, as well as a Qwen VLM fine-tuned on our training data. Results suggest that our dataset could improve performance of state-of-the-art models on downstream computer-use tasks.

1. Introduction

Recent advances in Large Language Models (LLMs) and Vision-Language Models (VLMs) have substantially improved the reasoning capabilities required for agentic applications (Zhou et al.; Andreux et al.; Wang et al.). Many real-world software systems and websites expose neither a public API nor a Model Context Protocol (MCP) endpoint, making the graphical user interface the only available control surface. Thus, visual grounding - the task of mapping a natural language instruction to an actionable screen region - plays a central role in the development of autonomous agents capable of performing complex tasks.

[1]H Company. Correspondence to: Nathan Bout <nathan.bout@hcompany.ai>, Ronan Riochet <ronan.riochet@hcompany.ai>.

Published as a workshop paper at SCALE - 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

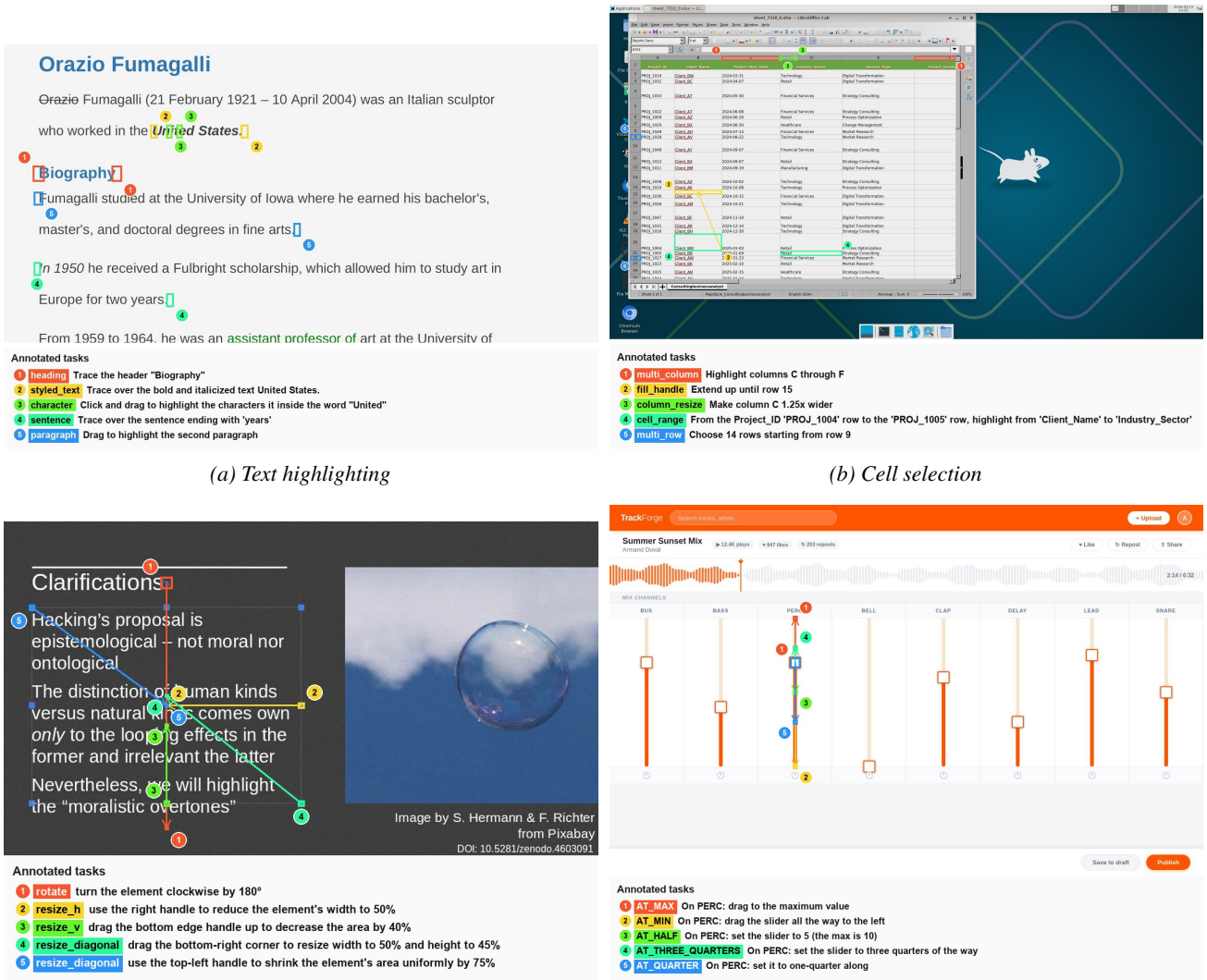
Early work on visual grounding has focused mainly on click grounding, which refers to a VLM ability to output a single coordinate (typically to execute a click action in a GUI). Click grounding has benefited substantially from scale: OS-ATLAS (Wu et al., b) released over 13M GUI elements across five platforms, UGround (Gou et al.) assembled 10M elements over 1.3M screenshots, and ScreenSpot (Cheng et al.) established a canonical evaluation on which subsequent models show steady progress. In contrast, less focus has been paid to *drag grounding*, the ability to output a pair of coordinates (e.g. to execute swipes or drag-and-drop actions in a GUI). For drag grounding, available data has remained an order of magnitude smaller (Liao et al.; Hu et al.) and largely dominated by text highlighting tasks. As a result, we show that current VLMs continue to perform poorly on drag-based interactions that are pervasive in GUI workflows.

In this paper, we close this gap by providing a large-scale drag grounding benchmark and training sets. Our contributions are threefold:

Rendering-as-supervision. We formalize a principle for constructing GUI-grounding data by exploiting the renderer's (PDF, XLSX, PPTX, HTML) own geometry as the labeling function. This method yields annotations at a fraction of the cost of VLM- and OCR-based approaches, with fewer construction errors.

DragOn Dataset. We instantiate the principle on four heterogeneous drag grounding domains – text highlighting, cell selection, element resizing, and slider manipulation – yielding a unified corpus of 286K training screenshots and 3.5M tasks, plus a 2k-example held-out evaluation suite. The dataset is one to two orders of magnitude larger than prior drag corpora.

Benchmark and baselines. We evaluate proprietary (GPT, Claude, Gemini) and open-source (Qwen, Kimi, Holo) models, as well as a VLM finetuned on our training data. Frontier models all score below 30%, showing that drag grounding remains a challenging problem for current VLMs. We also fine-tune a Qwen3.5VL on our training set, outperforming all frontier models on this task and showing that additional data can help improve VLM performance on



(a) Text highlighting *(b) Cell selection*

(c) Element resizing *(d) Slider manipulation*

Figure 1. The four drag grounding action domains covered by the proposed DragOn benchmark. Each example pairs a screenshot with a natural-language intent; the task is to predict a source and target bounding box on the screenshot, with an `ordered` flag indicating whether drag direction is semantically meaningful.

drag actions.

2. Related Work

A growing body of work investigates general-purpose agents that perceive, reason about, and act within GUIs, treating visual interaction as a universal interface for autonomous computer-use.

Computer-Use Agents. MAI-UI (Zhou et al.) introduces a family of foundation GUI agents trained with a self-evolving data pipeline and an online reinforcement learning framework, reaching state-of-the-art results on grounding and mobile navigation benchmarks. Surfer 2 (Andreux et al.) proposes a unified, vision-only architecture that combines hierarchical context management, decoupled planning and execution, and self-verification to operate across web, desktop, and mobile environments without task-specific fine-tuning. UI-TARS-2 (Wang et al.) further advances native GUI agents through multi-turn reinforcement learning and a hybrid environment integrating file systems and terminals, demonstrating strong performance across GUI, game, and software engineering tasks.

Agent Benchmarks. To evaluate performance of these agents, several benchmarks have been proposed. OSWorld (Xie et al.) provides a real, executable computer environment spanning Ubuntu and Windows, with 369 open-domain tasks evaluated by execution-based scripts; it highlights a large gap between human and model performance,

attributed primarily to GUI grounding and operational knowledge. AndroidWorld (Rawles et al.) offers a fully functional Android environment with reward signals over 116 programmatic tasks across 20 real-world apps, parameterized to yield diverse natural-language variations of each task. WebVoyager (He et al.) introduces an end-to-end web agent benchmark built from real-world tasks across 15 popular websites, together with an automatic evaluation protocol leveraging a multimodal model as judge. While these benchmarks measure overall task success, they provide limited insight into the specific perception and action subtasks on which agents fail. Drag interactions appear recurrently on end-to-end benchmarks.

Indeed, we find that $50/361$ (13.9%) of OSWorld (Xie et al.) tasks and $96/116$ (82.8%) of AndroidWorld (Rawles et al.) tasks require at least one drag action in a successful trajectory. On desktop, these drags cluster in spreadsheet range selection, text highlighting, as well as element resizing. On mobile, drags are near ubiquitous because simply launching an app from the home screen typically requires a vertical swipe on the app drawer. They are also used as a general-purpose swipe primitive across a variety of apps: freehand drawing in browser canvases (e.g. `BrowserDraw`, `BrowserMultiply`), map interaction in OsmAnd (placing markers, editing tracks), and direct manipulation of UI controls such as the system brightness slider. Appendix A lists every affected trial. Figure 2 shows representative drags from each benchmark.

Grounding Datasets. A complementary line of work isolates the grounding subtask, namely localizing the screen element referred to by a natural language instruction, which is a prerequisite for any successful action. SeeClick (Cheng et al.) identifies grounding as a key bottleneck for vision-only GUI agents and introduces ScreenSpot, a grounding benchmark covering mobile, desktop, and web interfaces, together with a dedicated grounding pre-training procedure. OS-ATLAS (Wu et al., b) releases the largest open-source cross-platform grounding corpus to date, with over 13 million GUI elements collected across Windows, Linux, macOS, Android, and the web, and trains a foundation action model that substantially narrows the gap with proprietary VLMs on grounding and out-of-distribution tasks. UGround (Gou et al.) trains a universal visual grounding model on 10M GUI elements over 1.3M screenshots, showing that a pure-vision agent matches or exceeds systems relying on accessibility-tree input. ScreenSpot-Pro (Li et al., a) extends grounding evaluation to high-resolution professional applications with small targets and complex layouts, where existing models perform poorly. More recently, Aria-UI (Yang et al.) and GUI-Actor (Wu et al., a) have further advanced click grounding through pure-vision instruction tuning and a coordinate-free attention-based action head, respectively. These datasets and methods, however, focus almost exclusively on click-grounding and do not evaluate the continuous, drag-based interactions that we target in this work.

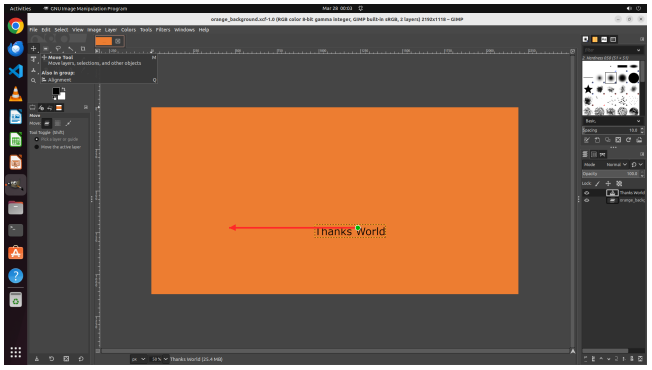
Programmatic and Synthetic Supervision. A complementary line of work constructs training data through structured pipelines rather than human annotation. Snorkel (Ratner et al.) formalizes weak supervision through user-written labeling functions whose noise is resolved by a generative model. Domain randomization (Tobin et al.) randomizes the renderer's parameters to diversify training inputs for sim-to-real transfer; its contribution lies in the input distribution rather than in the labeling channel. The closest precedents for our use of the renderer as an exact label source come from document understanding: DocBank (Li et al., b) extracts token-level bounding boxes from compiled LaTeX source, and Donut (Kim et al.) renders synthetic documents to produce paired image-text data at scale. We share their use of the renderer as oracle, but where these pipelines fix a single task per renderer (read all the text, extract all the layout boxes), our label map π is *query-conditional* and open-ended over substructures of S: the same (R, S) pair supports an arbitrary family of π at character, word, span, or layout granularity, parameterized by the natural-language query.

Drag Grounding. Two recent works target drag-based interactions specifically. (Liao et al.) introduce a training dataset and benchmark focusing exclusively on text selection. Our work subsumes text highlighting as one of four action domains and uses PDF coordinate extraction rather than OCR, yielding pixel-exact labels at arbitrary granularity - from single characters to multi-paragraph spans. (Hu et al.) propose a flow-based generative model for continuous drag trajectories, trained on 20K trajectories and evaluated on a 505-example benchmark split evenly across five application domains: PowerPoint, Adobe Premiere Pro, an OS file manager, handwriting, and captcha rotation. Their formulation targets closed-loop trajectory prediction whereas we focus on two-point bounding-box grounding from a static screenshot, a more tractable and faster-to-evaluate problem.

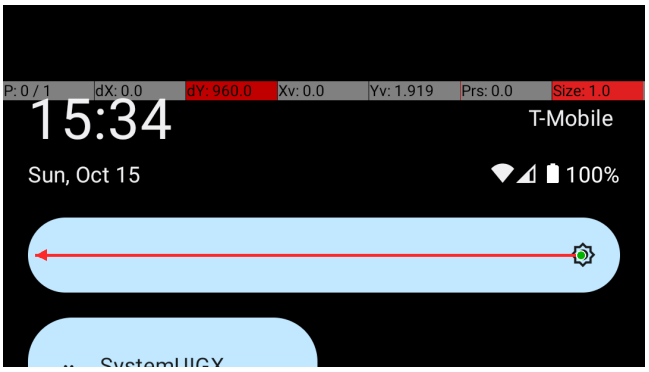
Relative to both prior works, our dataset is one to two orders of magnitude larger (see Table 1) and spans two unexplored domains: *cell selection*, and *slider manipulation*.

3. Rendering-as-Supervision

The four pipelines that produce *DragOn* share a common construction principle. Rather than annotating screenshots, we read ground-truth source and target bounding boxes directly from the renderer geometry. We refer to this principle as *rendering-as-supervision* and use it as a unifying frame for the dataset.



(a) OSWorld (`gimp_19`, step 2). Task instruction: *"Can you assist me in shifting the text box to the left? I keep accidentally selecting the image layer beneath it."* The agent issues `drag_and_drop(x1=0.55, y1=0.62, x2=0.35, y2=0.62)` to drag the 'Thanks World' text box leftward on the canvas.



(b) AndroidWorld (`SystemBrightnessMin-0`, step 2). Task instruction: *"Turn brightness to the min value."* The agent issues `mobile_drag(x1=0.9, y1=0.175, x2=0.05, y2=0.175)` to sweep the brightness slider down to the minimum. (Screenshot cropped to the top region containing the action.)

Figure 2. Representative drag actions from end-to-end agent benchmarks. The red arrow shows the executed drag overlaid on the observation the agent saw immediately before acting; drag endpoints are in normalized (x, y) screen coordinates.

Table 1. Scale and scope of GUI-grounding corpora. Our dataset brings click-grounding scale to drag, across four heterogeneous action domains.

Dataset	Modality	Train	Eval	Domains
OS-ATLAS (Wu et al., b)	click	13M	—	cross-platform
GUI-Drag (Liao et al.)	text drag	161K	5,333	1 (text)
ShowUI-π (Hu et al.)	drag trajectory	20K	—	5 apps
DragOn (ours)	drag grounding	3,5M	2,000	4 domains

3.1. Principle

We use R to denote a deterministic renderer that maps a structured source S to an image $I = R(S)$, and π to denote a *label map* that returns a pixel-space answer from a source and a query: $B = \pi(S, Q)$.

3.2. Analytic and Probe-Based Label Maps

The four DragOn domains (see section 4.1) use two flavors of π. When available, an *analytic* label map computes bounding boxes directly from the intent – for example, by composing PDF text-span coordinates from source, or by mapping PPTX EMU units to the pixel space of the rendered slide. Otherwise, we use *probe-based* label map that runs the renderer twice: a probe pass on a perturbed source (e.g. a cell flooded with a distinctive color key) is used to localize the target visually, after which the original source is re-rendered for the final screenshot. Probe-based label map is needed when no closed-form mapping is available but the rendered geometry can still be recovered to within rendering tolerance through visual detection on a controlled perturbation, which is the case in our cell selection domain (see section 4.1).

4. Benchmark

We create *DragOn*, a benchmark covering four types of complex drag actions: *text highlighting*, *cell selection*, *element resizing* and *slider manipulation*. In total it comprises 286K training screenshots and 3.5M training tasks, together with a 2,000-example held-out suite split evenly between a public validation set and a held-out test set (see Table 2). The four domains differ only in their instantiation of the (S, R, π) triple: PDF span coordinates, color-key pixel detection on a LibreOffice viewport, PPTX EMU units, and URL-parameterized HTML slider geometry, respectively. We summarize them below.

4.1. Domain Instantiations

Each of the four domains is generated through a dedicated pipeline that produces (i) a screenshot, (ii) a natural-language instruction, and (iii) the source and target bounding boxes defining the drag trajectory. Across all domains, bounding boxes are pixel-aligned with the rendered image either through analytic coordinate extraction or through probe-based visual detection, as described above. Light image augmentations (e.g. JPEG compression, blur, brightness

Table 2. Dataset statistics. Train counts reflect the unified benchmark after per-image task caps (10/26/12/10 for text/sheet/slide/slider) are applied to the upstream source corpora. Test and eval are disjoint held-out splits with one canonical intent per screenshot.

Domain	Train imgs	Train tasks	Test	Eval	Dominant resolution(s)
Text Highlighting	100,000	1,000,000	250	250	1275×1650 (portrait)
Cell Selection	38,332	996,632	250	250	1920×1080
Element Resizing	82,680	992,160	250	250	1280×720, 960×720, 960×540
Slider	65,000	571,350	250	250	1280×800
Total	**286,012**	**3,560,142**	**1,000**	**1,000**	—

and contrast jitter, small rotations) are applied to improve robustness.

Each instruction is sampled from a template that refers to the target by its: textual content, position (absolute or relative to another element), visual style, named entity, structural or semantic role (e.g. column header or a resize handle) or a quantitative specification (e.g. a rotation angle or a slider value). Most real instructions combine several of these. To ensure every instruction is answerable from the screenshot alone, we check referential uniqueness against the rendering before emission: ambiguous candidates are either scoped (*"in paragraph 3, select 'Federal' "*), disambiguated with an ordinal (*"the second 'Federal' on the page"*), or dropped. Appendix D shows example screenshots paired with the full set of intents sampled for each.

Text Highlighting. *S: styled DOCX from a Wikipedia article. R: headless LibreOffice DOCX-to-PDF. π: analytic, via PyMuPDF span coordinates and the page-to-image affine map.* We rely on Kaggle's structured Wikipedia dump as a static source of English and French articles. Each article section is parsed into plain text and rewritten as a styled DOCX document with varied fonts, colors, and inline highlights, so as to mimic the visual diversity of real-world documents. The DOCX files are converted to PDF using a headless LibreOffice instance, and the PDF is treated as the rendering source of truth for all downstream processing. Text selections of varying granularity (character, word, sentence, paragraph) are sampled with simple heuristics and combined with templated natural-language intents. Bounding boxes for the selected spans are obtained directly from PDF coordinate extraction, which yields exact pixel-aligned annotations on the rasterized image. Contrary to (Liao et al.), which relies on OCR to recover word-level boxes, extracting span coordinates directly from the PDF lets us vary the granularity at will, supporting tasks that require highlighting at the character level as well as at the paragraph level.

Cell Selection. *S: XLSX from a Pandas DataFrame. R: LibreOffice Calc in a Debian Docker image. π: probe-based, via color-key detection on a perturbed viewport.* Spreadsheet examples are fully synthetic. Realistic-looking tables, including calendars and gradebooks, are generated on-the-fly from Pandas DataFrames, exported to XLSX, and opened in a real LibreOffice Calc instance running inside a Debian Docker image. To recover pixel-accurate cell positions in the rendered viewport, we temporarily fill each cell with a distinctive color key, capture a screenshot, and locate cells by color matching, before reverting to the original styling for the final screenshot. Instructions are then templated from the detected cell grid and cover a range of interactions, including rectangular cell-range selections, multi-row and multi-column selections, column- and row-border resizes, and fill-handle drags (e.g. "extend formula down").

Element Resizing. *S: a slide from the Zenodo 10K PowerPoint corpus. R: headless LibreOffice slide-to-JPEG. π: analytic, via the EMU-to-pixel affine and per-handle offsets.* Slide examples are derived from the Zenodo 10K corpus of real-world PowerPoint presentations (DataCite & Make Data Count). Each deck is split into individual slides, which are rendered to JPEG via headless LibreOffice. Shape geometry is extracted directly from the underlying .pptx XML using `python-pptx` in native EMU units [1], and then mapped to the pixel space of the rendered slide. For each shape, we sample resize, rotate, and crop actions, each parameterized by one of the eight standard manipulation handles (four corners, four mid-edges, plus a rotation handle) and a target drag destination. Bounding boxes are computed geometrically from the shape's EMU bounding box and the corresponding handle offsets, so no visual detection is required. Annotated images additionally overlay the shape's selection border and handle circles, as illustrated in our training data.

Slider Manipulation. *S: a URL-parameterized HTML page. R: headless Playwright-Chromium. π: analytic, from value, track geometry, and per-variant thumb dimensions.* Slider examples are procedurally generated, with no external source data. We define six interactive HTML contexts, including a card dashboard, a media player, a settings panel, a photo editor, an audio mixer, and an e-commerce page, each instantiated in three visual variants. The full parameterization (color, theme, track size, thumb shape and size,

[1] English Metric Unit: A measurement in computer typography. There are 635 EMUs per twip, 6,350 EMUs per half-point, 12,700 EMUs per point, and 914,400 EMUs per inch. Used to translate on-screen layouts to printed layouts for specified printer hardware.

value range, step, endpoint labels, variant, etc.) yields a combinatorial space of approximately 850K unique page configurations.

For each URL, we compute which cues are actually visible (e.g. endpoint labels, tooltip, volume-icon convention with a 0–100 range) and sample instructions from a catalog of 17 templates restricted to those present in the rendering. This ensures every instruction can be answered from the screenshot alone. Pages are rendered with a headless Playwright-Chromium browser, captured at the slider initial value, and saved as JPEG. The source bounding box is derived from the thumb pixel position, computed from the initial and target values together with the track geometry and per-variant thumb dimensions; the target bounding box is a fixed 10×10 region around the drop point. Train, validation, and test splits are assigned deterministically by MD5 bucketing on the URL, ensuring that identical configurations consistently fall in the same split across regenerations.

4.2. Difficulty and Tolerance

Both resize and rotation are challenging tasks because the intent is highly precise (e.g. "increase the width of X by 30% of its current size" for resize, or "rotate X by 45 degrees clockwise" for rotation) and we set the target bounding-box dimensions to 5% of the element's current height and width (i.e. tolerance of $\pm2.5\%$). To better discriminate between models, we additionally report performance under a relaxed tolerance on the target bounding box: in the following, *acc@10%* and *acc@15%* denote success when the predicted point falls within a region scaled by 10% and 15% of the slide dimensions around the target, respectively.

4.3. Degrees of Freedom

For both resize and rotation, the set of acceptable target locations is in fact a continuum rather than a single point. When resizing along a single dimension using an edge handle, any point on the line orthogonal to that dimension passing through the target is equally valid. Similarly, for the 2D rotations considered in this work, any point on the half-line originating at the element center with the prescribed angle yields the same rotation. We illustrate the two conventions side-by-side in Figure 3 in the appendix.

For simplicity, we ignore these degrees of freedom and define the target bounding box canonically: for resize, on the axis aligned with the starting handle; for rotation, on the circle centered at the element center with radius equal to the distance from the starting handle to that center, on the same axis as the starting handle. Crucially, the training set is generated using the same canonical convention, preventing any spurious mismatch between training and evaluation targets.

4.4. Ordering

Some tasks require the click-and-drag to be executed in a specific direction, while others are direction-agnostic. In particular, cell selection and text highlighting accept either ordering of the two endpoints: dragging from the source to the target and dragging from the target to the source produce the same selection. In contrast, element resizing, rotation, and slider manipulation are inherently ordered, since the action is defined relative to the initial handle position. We expose this property explicitly via the `"ordered"` boolean field in the per-example JSON, which is used by the evaluation script. See figures 3a and 3b in appendix for an illustration.

4.5. Test Sets

We provide two held-out evaluation sets. The *validation set* is publicly released and intended for development and self-reported results. The *test set* is kept private and serves as the reference benchmark for cross-model comparison; participants may submit predictions and obtain their score, with optional inclusion on a public leaderboard. Both sets contain 1,000 examples, evenly distributed across the four task types (250 examples each), with a single canonical intent per screenshot.

5. Experiments

We evaluate a diverse set of models on our benchmark, spanning closed-source generalist agents accessed through public APIs and open-weight vision-language models served locally. We first describe the training procedure for our own model, then detail the evaluation protocol used for each baseline. All models are queried with structured (JSON) outputs encoding the four drag coordinates.

5.1. Training

We fine-tune a pretrained Qwen3.5-VL-35B-A3B (Qwen) model on our training dataset. We use Adam optimizer (Kingma & Ba) with learning rate `1e-6`, sequence length $20{,}480$, global batch size of 64 on 32 Nvidia H100 GPUs. We train the model for 1000 steps, corresponding to approximately 2 epochs.

5.2. Evaluation

We evaluate generalist and specialized models on the benchmark. For each baseline, we use the recommended settings provided in the documentation (e.g. decoding configuration, native coordinate convention), as described below:

Claude Sonnet 4.5. We query Claude Sonnet 4.5 (Anthropic) through the Anthropic API with extended thinking disabled, and obtain the four drag coordinates as a forced

tool call over a JSON schema, with coordinates expressed in image-pixel units.

Claude Opus 4.7. Claude Opus 4.7 (Anthropic) is queried in the same setting as Sonnet 4.5: Anthropic API, extended thinking disabled, structured output through a forced tool call, image-pixel coordinates.

GPT-5.4. GPT-5.4 (Singh et al.) is queried through the OpenAI API in JSON-object mode with default decoding, with coordinates expressed in image-pixel units.

Kimi-K2.5. Kimi-K2.5 (Team et al.) is queried through the Together AI API in JSON-object mode at temperature 0.6 and top-p 0.95 with reasoning disabled, and outputs coordinates as fractions of the image dimensions in $[0, 1]$.

Qwen3.5-VL and Holo3-35B-A3B. The Qwen3.5-VL-35B-A3B (Qwen) and Holo3-35B-A3B (HCompany) are served locally with vLLM in JSON-object mode with chain-of-thought mode enabled, and emit coordinates in the Qwen-native normalized $[0, 1000]$ frame. Qwen3.5-VL-122B-A10B and Qwen3.5-VL-397B-A17B are served in FP8 precision.

5.3. Results

Results on the test set are reported in Table 3, our model being denoted as *ours-35b-a3b*. The two strongest proprietary models, *claude-opus-4-7* (27.7%) and *gpt-5.4* (25.7%), perform reasonably well on slider manipulation (57.2% and 45.6%) and cell selection (37.2% and 28.0%), but their accuracy drops drastically on text highlighting (both below 10%) and element resizing (both below 20%). *claude-sonnet-4-5* is the weakest of the proprietary models on our benchmark, with an overall score of 10.7% and near-zero accuracy on the two text-based categories. Overall, no frontier model clears 30%, confirming that drag grounding is far from saturated even for the strongest closed systems.

Second, open-weight models trail proprietary frontier at matched or larger parameter counts. *qwen3-5-35b-a3b* collapses at 2.3%, *Kimi-K2.5* and *qwen3-5-122b-a10b-fp8* sit around 11%, and only the largest variant *qwen3-5-397b-a17b-fp8* (21.5%) approaches the proprietary band, still without reaching it. Computer-use specialization has a large effect at fixed architecture: compared to *qwen3-5-35b-a3b*, *holo3-35b-a3b* (which share the same 35B/A3B Mixture-of-Experts backbone) show an absolute performance boost of 20.9 points, close to proprietary frontier models.

Furthermore, we show that targeted training on our corpus pushes performance well past the frontier. Fine-tuning Qwen 35B/A3B base on our drag grounding data (*Ours-35b-a3b*) lifts overall success from 2.3% to 35.3%, a 33-point absolute improvement. Our model surpasses every evaluated frontier model by at least 7.6 points and takes the lead on three of the four action domains (text highlighting, slider manipulation, and element resizing), while remaining behind *claude-opus-4-7* on cell selection. These gains come entirely from data: the model class, size, and training pipeline otherwise match the open-source Qwen3-5 baseline.

Finally, a few limits of the current results are worth flagging. Element resizing remains the hardest domain in absolute terms, but the acc@10% and acc@15% columns suggest it is tolerance-limited rather than localization-limited: our model improves from 32.0% at the strict threshold to 46.8% at 10% and 57.2% at 15% of slide dimensions, with all frontier models moving in the same direction. This indicates that most remaining errors on resize are small-magnitude offsets around the correct handle rather than qualitative misgroundings.

5.4. Code

Code used to evaluate the models can be found in the following Github repository. Datasets can be found in the following HuggingFace repository, which will also include a leaderboard.

6. Conclusion

We presented a benchmark and training dataset for drag-based computer-use interactions, an aspect of GUI grounding that has so far received much less attention than click-grounding. The dataset spans four representative domains – *text highlighting*, *cell selection*, *element resizing*, and *slider manipulation* – and is built with task-specific pipelines that produce pixel-aligned source and target bounding boxes from structured sources, avoiding the noise of OCR- or detection-based annotation. In total, the release comprises 286K training screenshots and 3.5M training tasks, together with a 2,000-example held-out evaluation suite split into a public validation set and a private test set.

A comprehensive set of VLMs were evaluated on the benchmark: proprietary models (GPT, Claude), open-source VLMs (Qwen, Kimi, Holo), and an open-source VLM finetuned on our training data. Our results show that drag grounding remains challenging for current VLMs and that an open-source model fine-tuned on our dataset outperforms the strongest proprietary baselines, indicating that targeted training data could improve model performance on downstream computer-use tasks.

We hope that the benchmark, the dataset, and the public leaderboard will support further progress on drag grounding, a capability required for reliable computer-use agents.

Table 3. Success rate on the test set. The last two columns report success on element resizing under 10% and 15% spatial tolerance. Best result per column in **bold**.

	Success rate (%)					Tolerant ER (%)	
Model	**Total**	**Text Highlighting**	**Cell Selection**	**Slider Manipulation**	**Element Resizing**	**acc@10%**	**acc@15%**
Claude Sonnet 4.5	10.7	0.0	0.0	26.0	16.8	19.6	20.4
Claude Opus 4.7	27.7	7.6	**37.2**	57.2	8.8	13.6	16.0
GPT-5.4	25.7	9.2	28.0	45.6	20.0	23.2	26.4
Kimi-K2.5	11.6	12.4	14.0	16.0	4.0	5.2	7.6
Holo3 35B-A3B	23.2	20.8	34.4	14.4	23.2	30.0	36.4
Qwen3.5 35B-A3B	2.3	1.6	0.4	4.8	2.4	2.8	4.0
Qwen3.5 122B-A10B	10.8	6.4	7.2	19.6	10.0	14.8	18.8
Qwen3.5 397B-A17B	21.5	16.4	16.4	39.6	13.6	19.6	24.8
***Ours* 35B-A3B**	**35.3**	**33.2**	13.2	**62.8**	**32.0**	**46.8**	**57.2**

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

LLM and Agent Usage

We used large language models to proofread the manuscript. Coding agents assisted at various stages of dataset generation, particularly in creating the HTML for the slider selection domain. Figure 1 was also generated using a coding agent, based on actual images and annotations from our dataset.

References

Andreux, M., Bakler, M., Barbier, Y., Benchekroun, H., Biré, E., Bonnet, A., Bordie, R., Bout, N., Brunel, M., Cambray, A., Cedoz, P.-L., Chassang, A., Cloix, G., Connelly, E., Constantinou, A., Coster, R. D., Jonquiere, H. d. l., Delfosse, A., Delpit, M., Deprez, A., Derupti, A., Diaz, M., D'Souza, S., Dujardin, J., Edmund, A., Eickenberg, M., Fatalot, A., Felissi, W., Herring, I., Koegler, X., Kergaradec, E. L. J. d., Lac, A., Langevin, M., Lauverjat, C., Loison, A., Manevich, A., Moyal, A., Kerbel, A. N., Parovic, M., Revelle, J., Richard, G., Richter, M., Riochet, R., Santos, M., Savidan, R., Sifre, L., Theillard, M., Thibault, M., Valentini, I., Wu, T., et al. Surfer 2: The next generation of cross-platform computer use agents. URL http://arxiv.org/abs/2510.19949.

Anthropic. Claude 4. URL https://www.anthropic.com.

Cheng, K., Sun, Q., Chu, Y., Xu, F., Li, Y., Zhang, J., and Wu, Z. SeeClick: Harnessing GUI grounding for advanced visual GUI agents. URL http://arxiv.org/abs/2401.10935.

DataCite and Make Data Count. Data citation corpus data file. URL https://doi.org/10.5281/zenodo.14897662.

Gou, B., Wang, R., Zheng, B., Xie, Y., Chang, C., Shu, Y., Sun, H., and Su, Y. Navigating the digital world as humans do: Universal visual grounding for GUI agents. URL http://arxiv.org/abs/2410.05243. Issue: arXiv:2410.05243.

HCompany. Holo3 - open foundation models for navigation and computer use agents. URL https://huggingface.co/Hcompany/Holo3-35B-A3B.

He, H., Yao, W., Ma, K., Yu, W., Dai, Y., Zhang, H., Lan, Z., and Yu, D. WebVoyager: Building an end-to-end web agent with large multimodal models. URL http://arxiv.org/abs/2401.13919.

Hu, S., Lin, K. Q., and Shou, M. Z. ShowUI-pi: Flow-based generative models as GUI dexterous hands. URL http://arxiv.org/abs/2512.24965.

Kim, G., Hong, T., Yim, M., Nam, J., Park, J., Yim, J., Hwang, W., Yun, S., Han, D., and Park, S. OCR-free document understanding transformer. In *European Conference on Computer Vision (ECCV)*. URL http://arxiv.org/abs/2111.15664.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. URL http://arxiv.org/abs/1412.6980.

Li, K., Meng, Z., Lin, H., Luo, Z., Tian, Y., Ma, J., Huang, Z., and Chua, T.-S. ScreenSpot-pro: GUI grounding for professional high-resolution computer use, a. URL http://arxiv.org/abs/2504.07981.

Li, M., Xu, Y., Cui, L., Huang, S., Wei, F., Li, Z., and Zhou, M. DocBank: A benchmark dataset for document layout analysis. In *Proceedings of the 28th International*

Conference on Computational Linguistics (COLING), b. URL http://arxiv.org/abs/2006.01038.

Liao, Z., Lu, Y., Gou, B., Sun, H., and Awadallah, A. Beyond clicking: A step towards generalist GUI grounding via text dragging. URL http://arxiv.org/abs/2601.06031.

Qwen. Qwen3.5: Accelerating productivity with native multimodal agents. URL https://qwen.ai/blog?id=qwen3.5.

Ratner, A., Bach, S. H., Ehrenberg, H., Fries, J., Wu, S., and Ré, C. Snorkel: Rapid training data creation with weak supervision. 11(3). doi: 10.14778/3157794.3157797. URL http://arxiv.org/abs/1711.10160.

Rawles, C., Clinckemaillie, S., Chang, Y., Waltz, J., Lau, G., Fair, M., Li, A., Bishop, W., Li, W., Campbell-Ajala, F., Toyama, D., Berry, R., Tyamagundlu, D., Lillicrap, T., and Riva, O. AndroidWorld: A dynamic benchmarking environment for autonomous agents. URL http://arxiv.org/abs/2405.14573.

Singh, A., Fry, A., Perelman, A., Tart, A., Ganesh, A., El-Kishky, A., McLaughlin, A., Low, A., Ostrow, A. J., Ananthram, A., Nathan, A., Luo, A., Helyar, A., Madry, A., Efremov, A., Spyra, A., Baker-Whitcomb, A., Beutel, A., Karpenko, A., Makelov, A., Neitz, A., Wei, A., Barr, A., Kirchmeyer, A., Ivanov, A., Christakis, A., Gillespie, A., Tam, A., Bennett, A., Wan, A., Huang, A., Sandjideh, A. M., Yang, A., Kumar, A., Saraiva, A., Vallone, A., Gheorghe, A., Garcia, A. G., Braunstein, A., Liu, A., Schmidt, A., Mereskin, A., Mishchenko, A., Applebaum, A., Rogerson, A., Rajan, A., Wei, A., Kotha, A., Srivastava, A., Agrawal, A., et al. OpenAI GPT-5 system card. URL http://arxiv.org/abs/2601.03267.

Team, K., Bai, T., Bai, Y., Bao, Y., Cai, S. H., Cao, Y., Charles, Y., Che, H. S., Chen, C., Chen, G., Chen, H., Chen, J., Chen, J., Chen, J., Chen, J., Chen, K., Chen, L., Chen, R., Chen, X., Chen, Y., Chen, Y., Chen, Y., Chen, Y., Chen, Y., Chen, Y., Chen, Y., Chen, Y., Chen, Z., Chen, Z., Cheng, D., Chu, M., Cui, J., Deng, J., Diao, M., Ding, H., Dong, M., Dong, M., Dong, Y., Dong, Y., Du, A., Du, C., Du, D., Du, L., Du, Y., Fan, Y., Fang, S., Feng, Q., Feng, Y., Fu, G., Fu, K., et al. Kimi k2.5: Visual agentic intelligence. URL http://arxiv.org/abs/2602.02276.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. doi: 10.1109/IROS.2017.8202133. URL http://arxiv.org/abs/1703.06907.

Wang, H., Zou, H., Song, H., Feng, J., Fang, J., Lu, J., Liu, L., Luo, Q., Liang, S., Huang, S., Zhong, W., Ye, Y., Qin, Y., Xiong, Y., Song, Y., Wu, Z., Li, A., Li, B., Dun, C., Liu, C., Zan, D., Leng, F., Wang, H., Yu, H., Chen, H., Guo, H., Su, J., Huang, J., Shen, K., Shi, K., Yan, L., Zhao, P., Liu, P., Ye, Q., Zheng, R., Xin, S., Zhao, W. X., Heng, W., Huang, W., Wang, W., Qin, X., Lin, Y., Wu, Y., Chen, Z., Wang, Z., Zhong, B., Zhang, X., Li, X., Li, Y., Zhao, Z., et al. UI-TARS-2 technical report: Advancing GUI agent with multi-turn reinforcement learning. URL http://arxiv.org/abs/2509.02544.

Wu, Q., Cheng, K., Yang, R., Zhang, C., Yang, J., Jiang, H., Mu, J., Peng, B., Qiao, B., Tan, R., Qin, S., Liden, L., Lin, Q., Zhang, H., Zhang, T., Zhang, J., Zhang, D., and Gao, J. GUI-actor: Coordinate-free visual grounding for GUI agents, a. URL http://arxiv.org/abs/2506.03143. Issue: arXiv:2506.03143.

Wu, Z., Wu, Z., Xu, F., Wang, Y., Sun, Q., Jia, C., Cheng, K., Ding, Z., Chen, L., Liang, P. P., and Qiao, Y. OS-ATLAS: A foundation action model for generalist GUI agents, b. URL http://arxiv.org/abs/2410.23218. Issue: arXiv:2410.23218.

Xie, T., Zhang, D., Chen, J., Li, X., Zhao, S., Cao, R., Hua, T. J., Cheng, Z., Shin, D., Lei, F., Liu, Y., Xu, Y., Zhou, S., Savarese, S., Xiong, C., Zhong, V., and Yu, T. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. URL http://arxiv.org/abs/2404.07972.

Yang, Y., Wang, Y., Li, D., Luo, Z., Chen, B., Huang, C., and Li, J. Aria-UI: Visual grounding for GUI instructions. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 22418–22433. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.1152. URL https://aclanthology.org/2025.findings-acl.1152/.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*. URL http://arxiv.org/abs/2210.03629.

Zhou, H., Zhang, X., Tong, P., Zhang, J., Chen, L., Kong, Q., Cai, C., Liu, C., Wang, Y., Zhou, J., and Hoi, S. MAI-UI technical report: Real-world centric foundation GUI agents. URL http://arxiv.org/abs/2512.22047. version: 1.

A. Drag Actions in OSWorld and AndroidWorld

This section lists every task in which at least one drag is required to succeed, clustered by application domain.

A.1. OSWorld (50 / 361 tasks, 13.9%)

- GIMP (image editing) - 6 tasks
- LibreOffice Calc (spreadsheets) - 18 tasks
- LibreOffice Impress (slides) - 7 tasks
- LibreOffice Writer (word processing) - 9 tasks
- Multi-application workflows - 9 tasks
- VS Code - 1 task

A.2. AndroidWorld (96 / 116 tasks, 82.8%)

AndroidWorld tasks are clustered by target application.

- Notes and documents (`Markor`) - 11 tasks
- Calendars and to-dos (`SimpleCalendar`, `Tasks`) - 20 tasks
- Recipes and expenses (`Broccoli`, `ProExpense`) - 18 tasks
- System settings (brightness, Wi-Fi, Bluetooth) - 14 tasks
- Media apps (`RetroMusic`, `VLC`, `AudioRecorder`, `Camera`, `SimpleDrawPro`, `OpenTracks`) - 14 tasks
- Communication (`SimpleSms`, `Contacts`) - 7 tasks
- Browser-based tasks (`Chrome`) - 3 tasks
- Maps (`OsmAnd`) - 3 tasks
- Clock and file management (`Clock`, `Files`, `Gallery`, `Notes`) - 6 tasks

B. Alternative Ground-Truth Target Regions for Resize and Rotation

As discussed in the main text, both resize and rotation actions admit a continuum of equally valid drop points. In our benchmark we use a *canonical* ground-truth target - a small (5% of shape) axis-aligned box at a single representative endpoint - for both training data generation and evaluation. An *alternative* convention would be to make the ground-truth target cover the full set of equivalent drop points: for resize, a thin line orthogonal to the manipulated axis and passing through the target; for rotation, a half-line from the element's center at the prescribed angle. Figure 3 contrasts the two choices on a resize and a rotation example. We stick to the canonical convention for consistency with the training distribution, but both conventions are defensible.

C. Qualitative End-to-End Impact

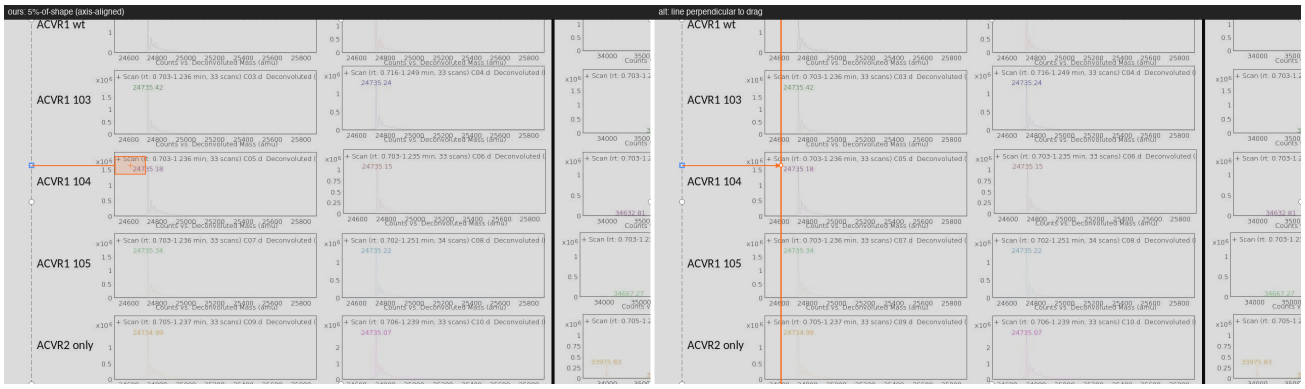
To illustrate how drag grounding translates to end-to-end computer-use performance, Figure 4 contrasts two agents on the same OSWorld task (`libreoffice_calc_19`): the task is to fill the *Gross Profit* formula down cells `J2:J10` of a spreadsheet. Both agents run the same ReAct-style scaffold (Yao et al.): at each step, the agent policy observes a screenshot, reasons about the current state in natural language, and emits an atomic action (click, drag, or keystroke) that is executed in the environment; the resulting screenshot is then fed back into the next step, until the task is completed or a step budget is exhausted. The scaffold, the prompt, the tool set, the screen resolution, and the step budget are held fixed across the two runs; the only difference between them is the underlying policy.

D. Drag-and-Drop Intents: Examples

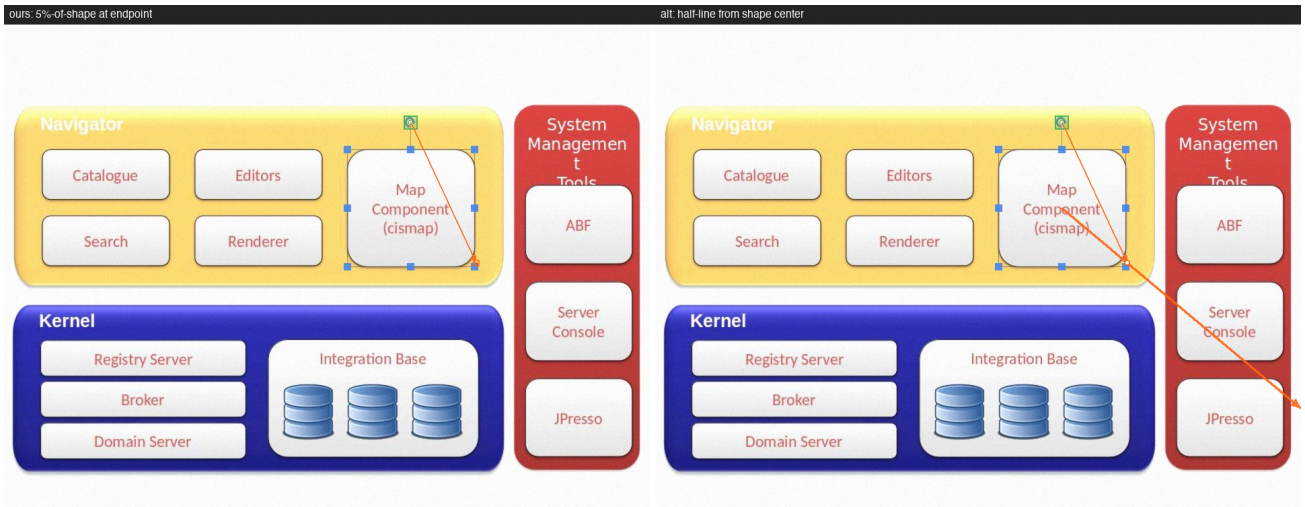
This appendix shows example screenshots from our benchmark together with the list of drag-and-drop intents collected for each one.

Intents for Figure 5:

- Highlight the range of the characters "cem" inside the word December



(a) Resize along the horizontal axis. **Left:** our canonical target, a 5%-of-shape axis-aligned box at the endpoint that matches the starting handle. **Right:** the alternative, a thin line perpendicular to the manipulated axis through the target, covering all drop points that yield the same resize outcome.



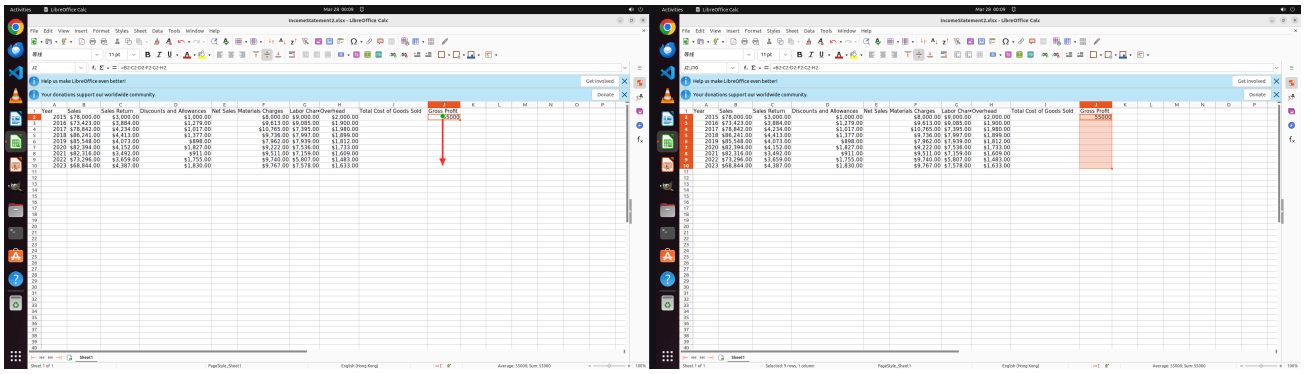
(b) Rotation. **Left:** our canonical target, a 5%-of-shape box at the endpoint on the circle centered at the shape's center. **Right:** the alternative, a half-line from the shape's center at the prescribed angle, covering all drop points that yield the same rotation.

Figure 3. Canonical vs. alternative ground-truth target regions for actions with a continuum of valid drop points. We adopt the canonical convention (*ours*, left in each pair) throughout the paper.

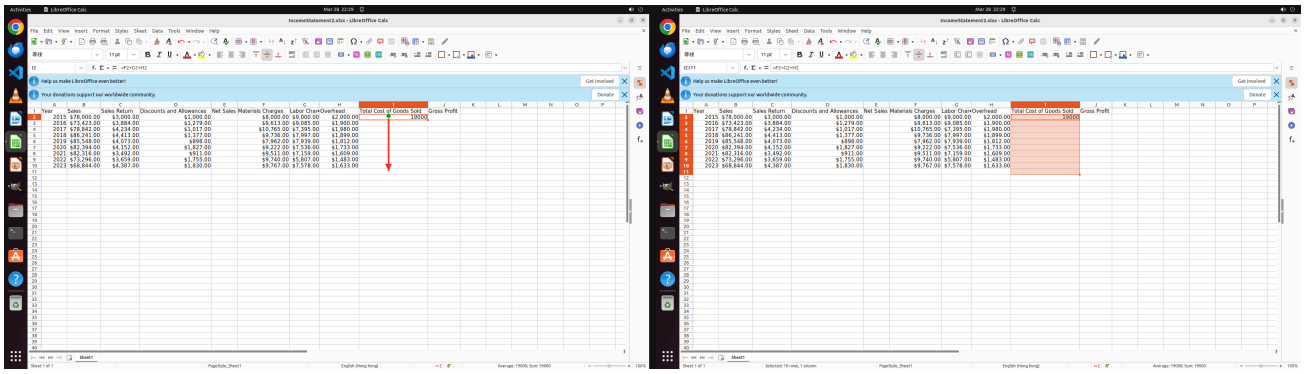
- Highlight the range of the text 'active Federal'
- Drag across the institution: 'Army Commendation Medals'
- Mark the extent of the paragraph starting with 'Location of headquarters changed'
- Trace 3 paragraphs starting from paragraph 2
- Highlight the range of the title
- Mark the extent of the text home stations; released from
- Find the organization 'Federal' and drag to highlight it in paragraph 3
- Click and drag to select the entire sentence starting with Organized and Federally recognized
- Swipe to select the word 2008 in the eighth paragraph

Intents for Figure 6:

- Mark the extent of the word 'for' on the second line
- Please select from start to end of paragraph 2
- Select act within 'abstract'



(a) **Successful drag - computer-use-specialized policy (`holo3-35b-a3b`).** Left: observation before the drag, with the executed drag call overlaid in red. Right: next observation, in which the agent has produced a valid drag that selects the required range of cells; the trial ultimately succeeds.



(b) **Failed drag - generalist base policy (`qwen3-5-35b-a3b-fp8`).** Left: observation before the drag, with the executed drag call overlaid in red. Right: next observation, in which the agent has not produced a valid drag, selecting the wrong range of cells; the trial ultimately fails.

Figure 4. Qualitative end-to-end comparison on the OSWorld task `libreoffice_calc_19`: the computer-use-specialized policy (Figure 4a) executes a successful drag and solves the task, while the generalist base policy with the same architecture and parameter count (Figure 4b) produces a failed drag on the same initial state; see Section C for the full setup.

- Draw a selection over ‘positioned in 11-space as’
- Outline the date ‘hendeca-10-tope’
- Highlight the 3rd ‘the’ on line 5
- Swipe to select the text ‘of (0,0,0,0,0,0,0,0,0,0,1). This construction is’
- Drag to highlight ‘faces, 330 6-simplex’
- Select the span of the sentence that ends with “9-faces”
- Trace over the header ‘Related polytopes’

Intents for Figure 7:

- Outline the word “Parliament” in the seventh paragraph
- Drag across ‘MP for’
- Click and drag to select the paragraph that begins ‘The Liberal Party candidate’
- In paragraph 7, drag across the last sentence
- Select “te” within “candidate” in paragraph 5
- Trace ‘Parliament’ in paragraph 2
- Highlight the range of the heading starting with ‘Result’
- Select the person’s name: ‘William Pease’ in paragraph 4

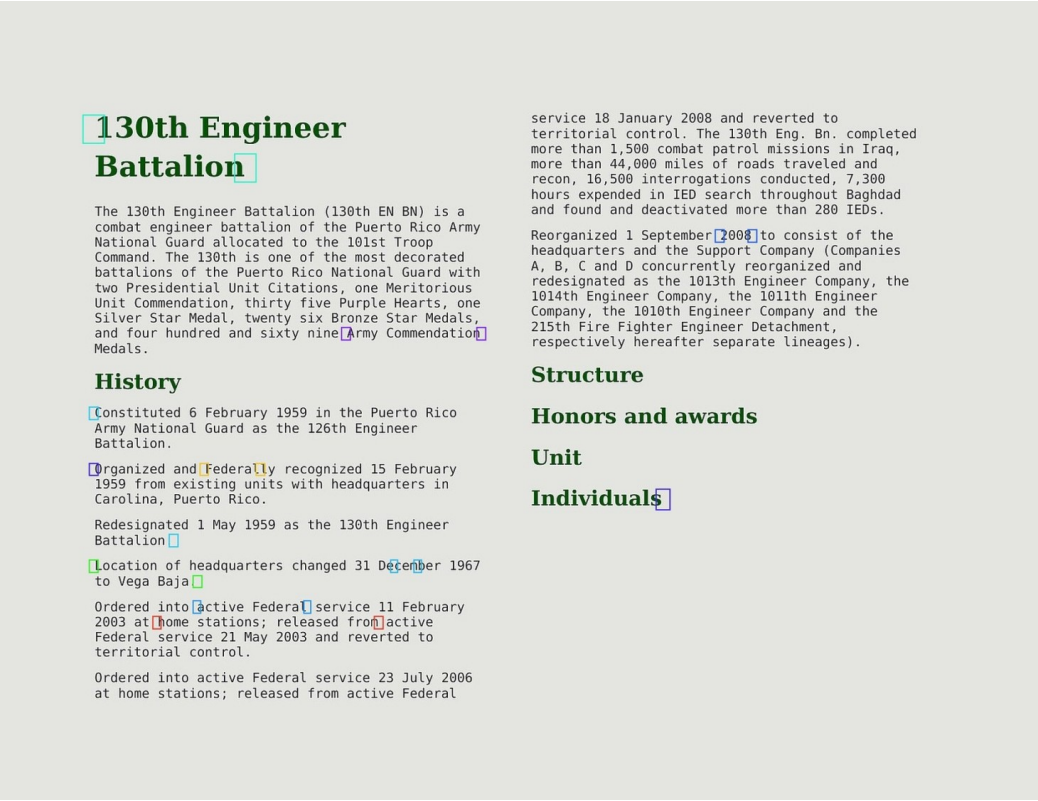


Figure 5. Text highlighting example.

- Drag to select the text his defeat at the

Intents for Figure 8:

- Resize the 3rd column to be 0.5x its current width
- Resize the 8th row to be 1.75x its current height
- Drag to select from the cell containing ‘2026-04-07’ to the cell containing ‘9:00’
- Choose from the ‘Duration_Hrs’ column to the ‘Room’ column
- Use the fill handle at G19 to extend down by 4
- Resize row 5 to be 0.5x its current height
- Use the fill handle at G19 to extend down by 2
- Drag to extend to the ‘2026-03-24’ cell to the left
- Drag to select from F7 to G10
- Drag from the Date column header to the Duration_Hrs column header
- Select the range from ‘Review’ to ‘2026-04-11’
- Mark columns ‘Start_Time’ to ‘Duration_Hrs’ across rows 22 to 23
- Make row 18 0.7x shorter
- Mark from the ‘Person_10’ cell in the Organizer column to the Start_Time cell on that row
- Select from ‘Date’ to ‘Room’, rows 19–25
- Select the next 6 rows beginning at row 18
- Highlight 3 columns starting from column D
- Highlight cell E3 and the 3 cells below
- Mark the block spanning rows 15–16 and columns D–E

- Highlight the range from ‘2026-03-21’ to ‘3’
- Select the range C21:D23
- Drag from ‘2026-04-05’ to ‘Person_10’
- Select 4 rows starting from row 11
- Drag from row 5 to row 8
- Choose columns ‘Event’ to ‘Organizer’ across rows 2 to 18
- Mark from the 3rd ‘Room A’ to the 3rd ‘Conference’ in the Room column
- Mark 14 rows starting from row 12
- Highlight cell H9 and the 3 cells below
- Extend up until row 15
- Use the fill handle at G19 to extend right by 1
- Resize row 9 to fit 10 characters of wrapped text
- Make the 6th column 0.4x narrower
- Select from ‘Start_Time’ to ‘Event’, rows 22–23
- Drag the fill handle 2 cells left
- Highlight from the cell containing ‘Review’ to the cell containing ‘2026-03-25’
- Resize the ‘Start_Time’ column to be 2x its current width
- Highlight from the cell containing ‘15:00’ to the cell containing ‘2026-03-20’
- Make row 20 3x taller
- Choose from C1 to G24
- Extend down until row 24
- Drag the fill handle 2 cells up
- Highlight cells F11 through G14
- Select from the 1st occurrence of ‘5’ in the Attendees column to the Duration_Hrs cell on that row
- Extend the Room fill handle 1 row below
- Drag the fill handle right to column I
- Make the 1st column 0.4x narrower
- Drag to select the next 4 columns beginning at column F
- Widen the ‘Room’ column by 2x
- Select columns E to G
- Drag from column E to column H

Intents for Figure 9:

- drag the bottom-right crop corner to crop the visible width by 40% and cut height by 10%
- trim the visible area by 25% using the bottom-left crop corner
- trim the visible area by 50% by dragging the right crop handle left
- crop the visible area by 15% by dragging the top crop handle down
- pull the right crop handle left to trim the width to 80%
- pull the bottom-left crop corner to crop the visible area by 19%
- trim the visible area by 30% using the bottom crop handle
- drag the top crop handle down to cut the area by 35%
- cut the visible width by 25% via the left crop handle
- crop the visible area proportionally to 30% using the top-left crop corner
- trim the visible area to 56% by dragging the top-left crop corner
- drag the bottom-right crop corner to cut width to 25% and cut height to 50%
- cut the area to 50% by pulling the top crop handle down

- drag the left crop mark right to reach 75% visible area
- cut the area by 75% by pulling the right crop handle left
- trim the visible area proportionally by 88% using the bottom-right crop corner
- trim the area proportionally to 64% by dragging the top-left crop corner
- use the top-right crop handle to set visible width to 75% and height to 50%
- use the top-right crop handle to set visible width to 55% and height to 95%
- trim the area by 60% via the bottom crop handle up

Intents for Figure 10:

- Drag to the minimum value
- Knowing the price max is $1000, drag to USD 800
- The slider runs 0 to $1000; set the price to 700 dollars
- On a 0 to $1000 scale, set the price to 50 dollars
- The slider runs 0 to $1000; set the price to 850 dollars
- On a 0 to $1000 scale, set the price to 100 dollars
- The price scale goes up to $1000; set it to 150 dollars
- Put the slider at the three-quarter point
- Set the price to USD 650 (the max is $1000)
- Set to $1000 (the maximum)

Intents for Figure 11:

- On SNARE: drag to the highest position
- On SNARE: set the volume to 100
- On SNARE: move the volume to 80
- On SNARE: put the volume at 40
- On SNARE: set to 0 (the minimum)
- On SNARE: change the volume to 60
- On SNARE: the volume ranges from 0 to 200; put it at 160
- On SNARE: change the volume to 20
- On SNARE: the slider runs 0 to 200; set the volume to 120

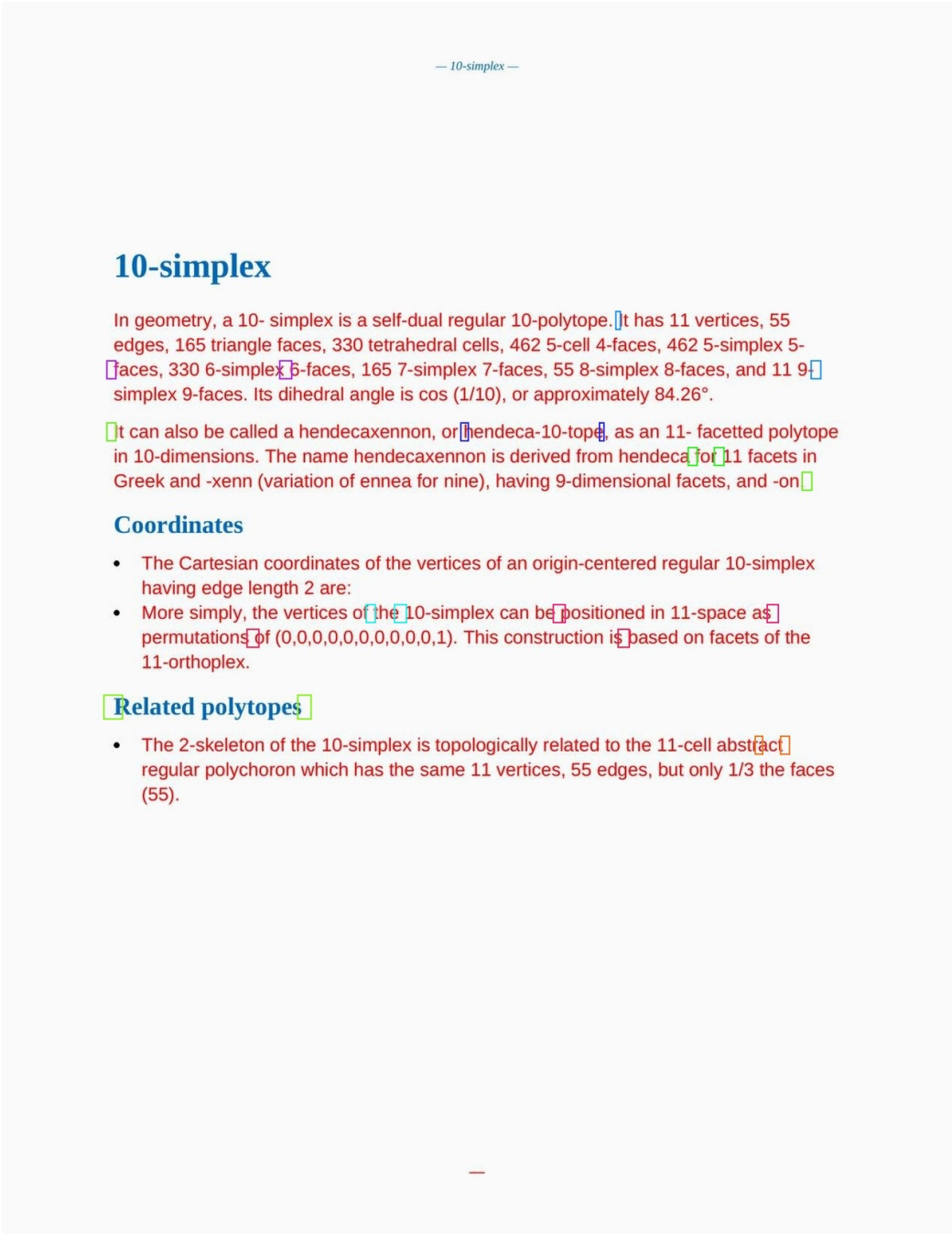


Figure 6. Text highlighting example.

Figure 7. Text highlighting example.

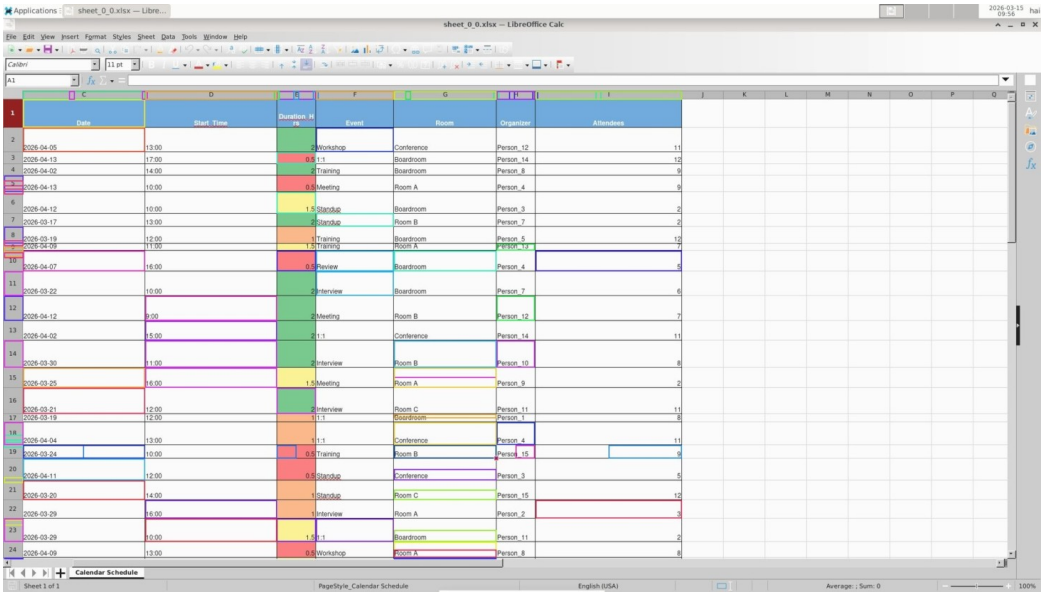


Figure 8. Cell selection example.

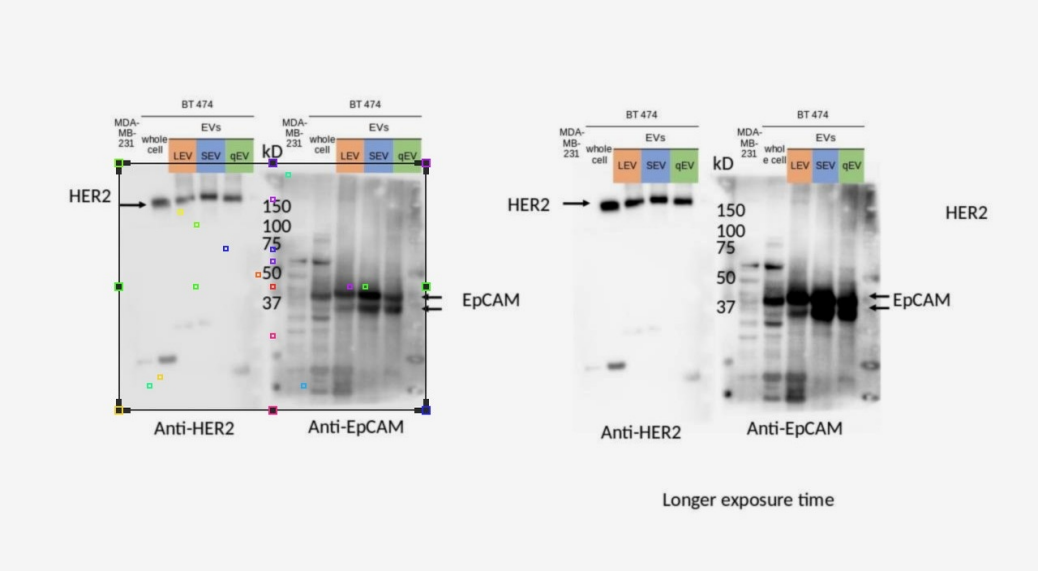


Figure 9. Element resizing example (crop).

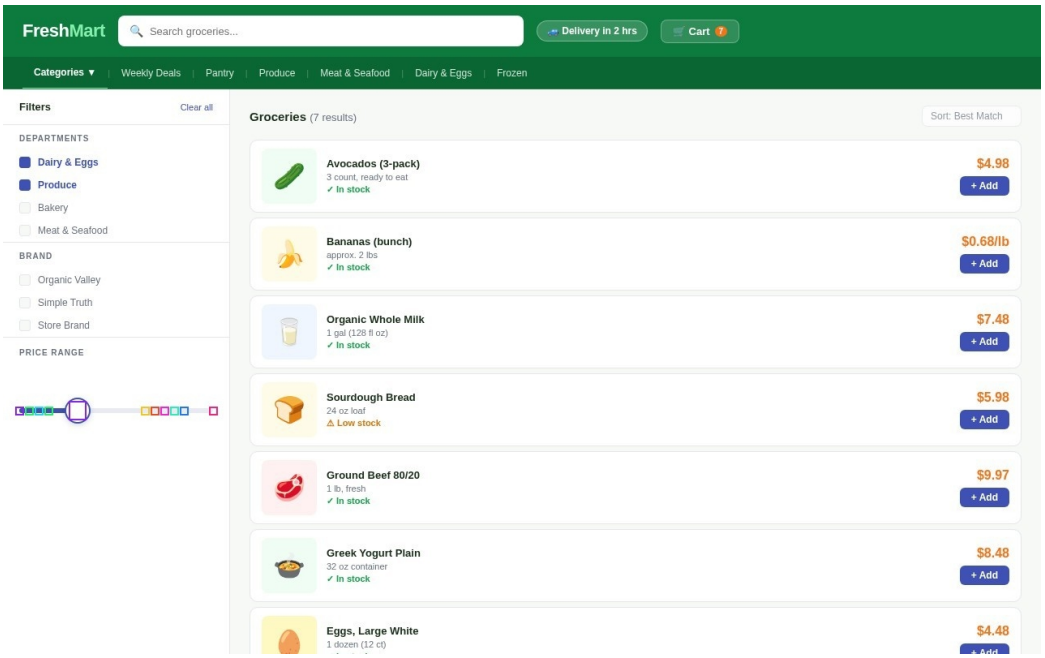


Figure 10. Slider manipulation example.

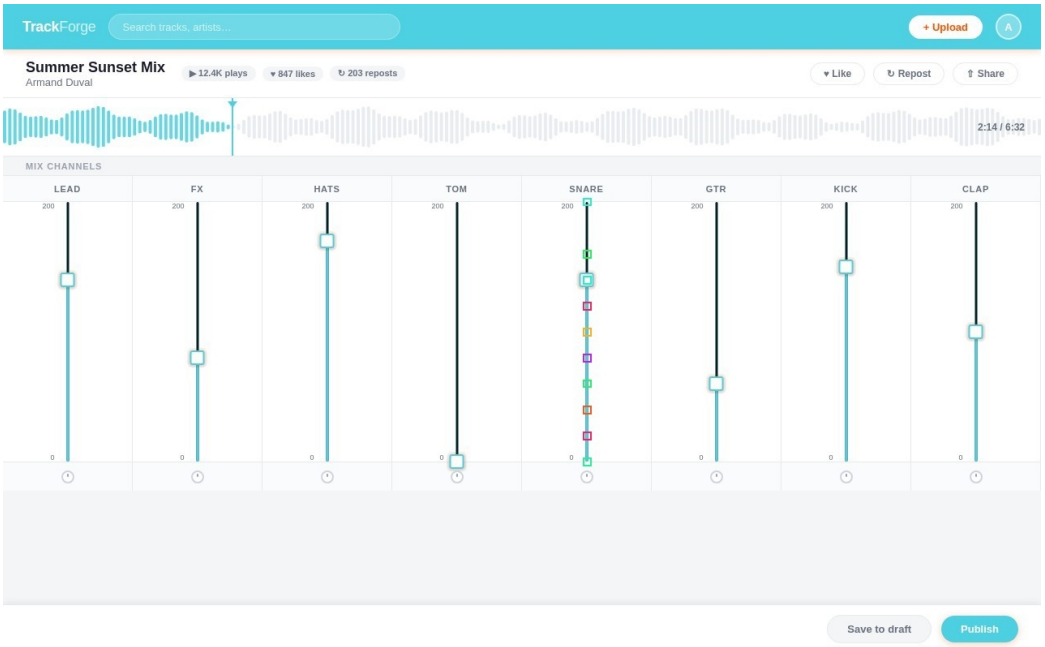


Figure 11. Slider manipulation example (vertical mixer fader).