KEEPING RECORDED GUI WORKFLOWS RELIABLE WHEN INTERFACES CHANGE

Anonymous authors
Paper under double-blind review

ABSTRACT

Conventional GUI record and replay is fast and cheap, but it breaks under interface reflow, restyling, distractors, and renamed controls. Vision-language computer-use agents adapt to such changes, yet re-planning an already demonstrated workflow on every run adds latency, cost, and error. We study the middle ground: given one successful demonstration, keep that workflow reliable under interface perturbations without generating a new plan. Our method preserves the recorded action sequence and re-ground each target from pixels on the live interface. Our resolver ranks OCR, template, and geometry candidates with 82 scene-relative features and uses calibrated confidence to blend learned evidence with the Hybrid prior and to trigger bounded recovery. We trained on a synthetic corpus of 38 designed workflows rendered under identity with 9 perturbation classes and evaluated on three benchmarks covering 464 replayable workflows from MiniWoB++, WebArena/WONDERBREAD, and WebShop under identity and six task-preserving perturbation classes, yielding 2,670 admitted task–perturbation cells and 16,020 method episodes. Averaged over the six perturbation classes, our method obtains 88.4% MiniWoB++ success versus 67.2% for Hybrid, and 83.9% WebArena success versus 55.2%. On WebShop it reaches 97.6 reward points against 96.3. Code is available at: `https://anonymous.4open.science/r/ANONYMIZED`.

1 INTRODUCTION

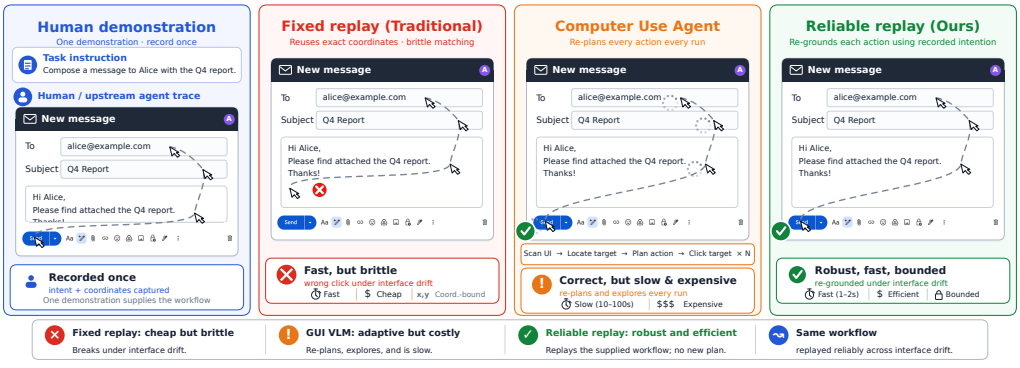


Figure 1: Upstream demonstrates the workflow once (left). With a slight GUI change, fixed replay clicks the wrong control since references drift. A computer-use agent completes the task with expensive and delay. Our solution keeps replay from a single demonstration reliable and efficient.

The same GUI task can be different at the pixel level due to the change of font-size, spacing, pop-out distractors. However, day-to-day workflows are often repetitive. Recording one successful execution and replaying it is attractive because the repeated run is cheap, deterministic, and easy to inspect. Conventional automation therefore remains useful even when a more general interactive system is available: once a valid procedure is known, repeated planning is unnecessary. Record-and-replay systems, from CoScripter to robotic process automation, get the procedure for free from the user and freeze the references at record time (Leshed et al., 2008; van der Aalst et al., 2018). A demonstration

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

actually bundles two conceptual properties: 1. *procedure*: which actions, in which order, with which typed values to execute to complete the task. It is exactly what a computer-use agent spends its per-step reasoning on. Procedure is relatively stable, because interface perturbations do not change what the user is trying to do. 2. *references*: the coordinates, target crops, and visible labels that determine where each action lands. Reference is cheap to obtain but fragile. Agent frequently missed the correct references. Record-and-replay holds both fixed, so a single invalidated reference fails the entire run. A agent treats both as unknown, re-deriving the procedure it was already given therefore the frontier-model cost on every step of every run. Neither pays in proportion to what actually changed. Procedure failures are a planning problem and belong upstream, with the human or agent that supplies the demonstration; we take the demonstrated procedure as correct and hold it fixed. In this paper we only studied the reference failures and solutions.

Document Object Model (DOM), as the mature reference on the web, has been widely researched, implemented and production ready Leotta et al. (2016); Choudhary et al. (2011); Leotta et al. (2015); Nass et al. (2023). While the web reference took a big portion of the GUI tasks, control over desktop, gaming where user does not have access for DOM become challenge for above solutions. Different approaches tried to resolve this problem from pixel level by text (Tex.) Coppola et al. (2025), template (Temp.) Yeh et al. (2009); Alégroth et al. (2015); Stocco et al. (2018), geometry (Geo.) matching Nass et al. (2023), and also combine all three as *Hybrid* (Hyb.) method Sereshkeh et al. (2020); Yu et al. (2021). However, different task with difference GUI will not depend on same setting of hard-coded design. Implement heavy weight on text (OCR) over GUI without text will under perform the performance while same setting could performed well under rich text screenshot. In this paper,

1. We propose a lightweight, pixel-based resolver trained on synthetic workflows that generalizes to unseen tasks and applications. Given a single upstream demonstration, it enables reliable and efficient replay without requiring a generative model at execution time.
2. We construct 38 real-world-style workflows containing more than 500 demonstrated actions across 9 synthetic perturbation classes for model training and then evaluate on 464 replayable workflows from 3 benchmarks across 6 interface perturbation classes, yielding 2,670 task–perturbation cells and 16,020 method episodes.
3. We then studied on (i) performance on our solution. (ii) the correlation between task length and task success, (iii) a controlled experiment to test adaptability when each perturbation group is omitted from training, and (iv) the ablation study on reliability over DOM-based references.

Across the 6 perturbation classes, learned replay reaches 88.4% task success on MiniWoB++ and 83.9% on WebArena, compared with 67.2% and 55.2% for Hybrid. On WebShop, it obtains a mean native reward of 97.6, compared with 96.3 for Hybrid. The advantage of our solution grows with action sequence which previous work did not addressed properly: on WebArena demonstrations of seven to fifteen actions, learned replay holds 83.6% task success while Hybrid falls to 50.7%.

2 Related Work

Procedure Failures and Reference Failures A GUI task can fail in two ways. In a *procedure failure*, the workflow is incomplete or incorrect. In a *reference failure*, the procedure remains correct, but an action targets the wrong control because the interface has changed. Procedure failures concern *what* to do. Language-model agents address them through per-step reasoning, but often redundantly re-derive a procedure already supplied. Computer-use research similarly separates planning from grounding (Cheng et al., 2024; Gou et al., 2024). Reference failures concern *where* to act and are more common. Page objects isolate locators from test logic, reducing repair time threefold and modified code eightfold (Leotta et al., 2013). Likewise, 73.62% of 1,065 test breakages arose from stale element locators rather than incorrect procedures (Hammoudi et al., 2016). Agents suffer the same failure when visual changes cause mis-clicks without changing the task (Wang et al., 2026).

References from Page Structure DOM-based reference is a relatively mature and widely deployed technology. Web elements are identified through well-defined symbolic queries, such as XPath expressions, CSS selectors, and predicates over element attributes. These references underpin everyday tools for testing, scraping, and browser automation. Extensive research has further improved their robustness through resilient locator generation, cross-version repair, and attribute-based element

Table 1: *Reference* only methods comparison from related works over features. Horizon denotes the evaluated sequence length: *I*=1–3, *II*=4–6, *III*=7–15, and *IV*=$\geq$ 16

Method	Depends on		Evidence			Sequential evaluation	
	DOM	Pixels	Text	Template	Geometry	Horizon	Length analysis
LTR (Coppola et al., 2025)	✓	✓	✓	✗	✓	–	✗
Similo (Nass et al., 2023)	✓	✓	✓	✗	✓	–	✗
ViSTA (Stocco et al., 2018)	✓	✓	✗	✓	✗	IV	✗
LIRAT (Yu et al., 2021)	✗	✓	✓	✓	✓	-	✗
VASTA (Sereshkeh et al., 2020)	✗	✓	✓	✓	✓	II–III	✓
Sikuli (Yeh et al., 2009)	✗	✓	✗	✓	✗	–	✗
Reliable Replay (ours)	✗	✓	✓	✓	✓	I–IV	✓

matching (Leotta et al., 2016; Choudhary et al., 2011; Leotta et al., 2015; Nass et al., 2023; Coppola et al., 2025). Even screenshot-based methods ultimately map visual matches back to DOM elements (Stocco et al., 2018). Its limitation is availability. Desktop applications, games, canvas-rendered interfaces, remote sessions, and some embedded webviews expose pixels but no usable DOM. Modern web features such as hashed classes, shadow roots, and canvas rendering can also weaken structural references. User desired programmatic automation based on DOM could also be restrict through authentication, bot detection. Thus, although DOM referencing is well defined, extensively studied, and industrially established, it does not address settings where page structure is hidden or absent—the setting considered in this paper.

References from Pixels Though DOM with limit, resolving references without page structure is not new. Pixel-based referencing has developed in three waves. **1.** Visual GUI testing Alégroth et al. (2015), beginning with Sikuli, locates controls by matching recorded image templates; it is practical but costly to maintain under restyling and resolution changes with human involve in coding on *procedures* (Yeh et al., 2009). **2.** Parse-then-match systems first recover controls through detection and OCR, then match references using appearance, text, and layout. VASTA checks the demonstrated location, template, and falls back to detection and OCR, without using a UI hierarchy (Sereshkeh et al., 2020) but it did not resolve the long sequence issue as single step failure would cause the whole sequence failure. LIRAT use screenshot matching with an OCR-assisted parsing and layout reconstruction and arbitrates the two using a fixed rule (Yu et al., 2021) but only focused on single step. Modern parsers such as OmniParser supply the same inventory at much higher quality (Lu et al., 2024). Commercial vision-based automation follows the same recipe with multi-anchor descriptors and a hand-set acceptance threshold. In the **3rd** wave, grounding models map a language description and screenshot directly to coordinates (Cheng et al., 2024; Gou et al., 2024; Qin et al., 2025), this is what current computer-use agents rely on, but require model inference and a textual query at every step. The central difficulty is combining visual, textual, and positional evidence. Earlier systems use fixed weights, fallback rules, or thresholds, although the most reliable signal depends on the interface change: restyling alters appearance, reflow alters position, and renaming alters text. Existing evaluations mainly measure image maintenance effort over release histories; parse-then-match systems report element-level matching accuracy, or end-to-end success on task suites of their own construction. They do not measures end-to-end task success on public benchmarks under interface changes that are applied deliberately and leave the task unchanged, which is what isolating reference resolution requires.

3 METHOD

Problem Contract, Learned Ranking, and Confidence Gating A workflow u supplies an initial state, a native terminal evaluator E_u, and one successful demonstration $D_u = ((o_1, a_1), \ldots, (o_T, a_T))$. At step t, the resolver receives the demonstrated reference o_t and the current screenshot, constructs candidates $\mathcal{C}_t$, and represents candidate i by a fixed feature vector $\mathbf{x}_{ti} \in \mathbb{R}^{82}$. Its entries derive from three evidence channels—text similarity between the demonstrated reference and live OCR, global and local template correlation, and a relative geometry prior—from

within-scene statistics of those channels (fractional and reciprocal rank, margin to the runner-up, and z-score across $\mathcal{C}_t$), and from channel-availability indicators, candidate-source flags, and broadcast scene context. The within-scene statistics are computed from the fixed evidence channels alone: they record how candidate i compares with its competitors at step t, and involve neither the ranker's own output nor the oracle label. An absolute similarity of 0.7 means different things in a sparse dialog and a dense table, whereas a rank within $\mathcal{C}_t$ is comparable across scenes. Table 9 gives the full composition. The ranker estimates $p_{ti} = \sigma(F(\mathbf{x}_{ti}))$ and the policy selects $\hat{\imath}_t = \arg\max_i s_{ti}$, where s_{ti} is defined below, then applies the recorded action a_t at that candidate.

When the learned ranking has low confidence, the resolver first invokes a soft fallback rather than forcing its top prediction. A logistic calibrator fit entirely within the training corpus estimates $q_t = P(\text{model top-1 is correct} \mid \max_i p_{ti}, A_t)$, where A_t is the number of agreeing evidence channels. The deployed score is

$$s_{ti} = \lambda_t \widetilde{p}_{ti} + (1 - \lambda_t)\widetilde{h}_{ti}, \qquad \lambda_t = \mathrm{clip}(\lambda_0 q_t, 0, 1),$$

so low q_t shifts weight from the learned probability p_{ti} to the Hybrid prior h_{ti}. If q_t is below the training-selected threshold θ, the bounded-recovery harness may dismiss a recognized blocking overlay or scroll and rescan, and stops recovery after two rounds; it aborts only when no target location is returned.

Because workflow success does not identify the correct candidate at each step, we train the ranker with oracle candidate labels y_{ti}. As candidate generation produces far fewer positive rows than negative rows, we introduce a sample weight w_{ti} so that positive target evidence is not overwhelmed by the many negative candidates. Negative rows receive unit weight, while positive rows receive the training-set negative-to-positive ratio N_-/N_+, capped at 9 to avoid unstable updates from excessive up-weighting:

$$\mathcal{L}(F) = -\sum_{t,i} w_{ti}\big[y_{ti}\log p_{ti} + (1 - y_{ti})\log(1 - p_{ti})\big], \qquad w_{ti} = \begin{cases} \min(N_-/N_+, 9), & y_{ti} = 1, \\ 1, & y_{ti} = 0. \end{cases}$$

We use a monotonic histogram gradient-boosted tree because the reliability of text, template, and geometry changes through nonlinear feature interactions. Its update is $F_m = F_{m-1} + \eta f_m$; a minimum leaf size of 40 prevents splits supported by only a few candidates. Remaining estimator settings are given in Appendix D.

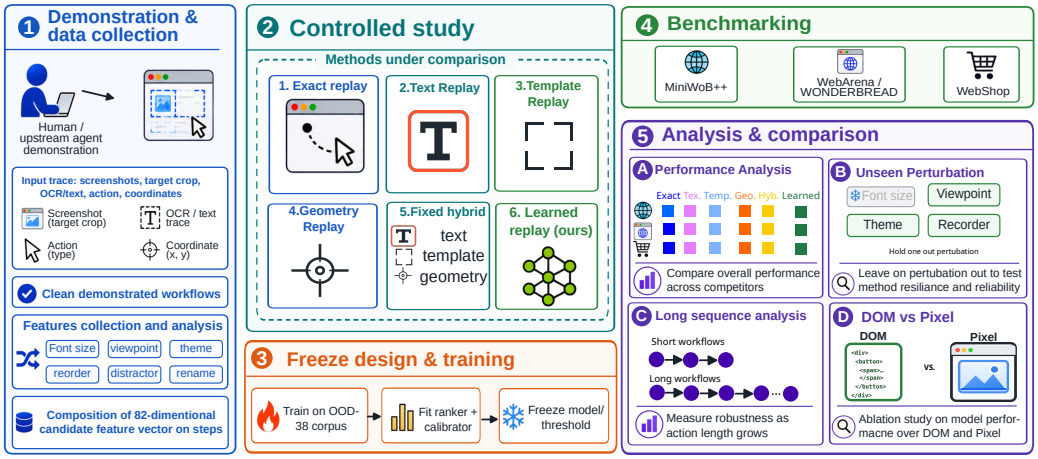


Figure 2: The experimental workflow.

Candidate Construction and Fixed Evidence For each demonstrated step, the hybrid and learned resolver constructs a bounded candidate set from three sources: every OCR text box returned for the live screenshot, the eight strongest non-overlapping normalized-cross-correlation peaks pooled across the full template scale grid 0.6–2.2 rather than taken at a single best scale, and one projection of the demonstrated target box into the live viewport. A safety cap of 80 text boxes per scene guards against

a pathological recognition result; it never binds in our collection, whose densest scene contains 77 boxes, so no candidate is discarded by the cap. Candidates whose intersection-over-union reaches 0.5, or whose centers lie within 12 pixels, are merged into one candidate that inherits the union of their source flags. The OCR and template sources provide content-driven proposals, while the projection preserves a candidate when only appearance changes; crucially, the ranker can select only from this union and cannot hallucinate an arbitrary screen coordinate.

Baselines are defined by the evidence a method consumes rather than by its implementation. Table 1 groups prior pixel-based replay into three families: matching recognized *text* against the demonstrated label (Sereshkeh et al., 2020), matching the recorded crop by *template* appearance (Yeh et al., 2009; Alégroth et al., 2015; Stocco et al., 2018), and reusing the recorded position as a *geometry* prior (Yu et al., 2021; Nass et al., 2023). We instantiate one channel per family,

$$e_{ti}^{\text{geometry}} = \exp\left[-\frac{1}{2}\left(\frac{\Delta x_{ti}^2}{(0.25W)^2} + \frac{\Delta y_{ti}^2}{(0.25H)^2}\right)\right],$$

measured from the viewport-projected demonstration center, together with e_{ti}^{text}, the normalized fuzzy match between demonstrated and live labels, and e_{ti}^{template}, the positive part of the best local multi-scale normalized cross-correlation. Each family is then the limiting case in which one channel's weight is one and the others are zero, and we implement each in the form used by its own literature: text matches the demonstrated label against the recognized text boxes, template takes the multi-scale correlation maximum over the whole screenshot, and geometry ranks the candidate set by position alone. Exact replay is reported alongside them because it executes the raw recorded coordinate without ranking candidates at all.

Single-channel rules fail in complementary ways, because each is defeated by the perturbation class that attacks the evidence it reads: renaming defeats text, restyling defeats template, and reflow defeats geometry. Since none can compensate with another channel, the natural non-learned baseline is not any one of them but their Hybrid combination,

$$h_{ti} = \frac{0.45e_{ti}^{\text{text}} + 0.35e_{ti}^{\text{template}} + 0.20e_{ti}^{\text{geometry}}}{0.45I_{\text{text}} + 0.35I_{\text{template}} + 0.20},$$

where an availability indicator removes a missing channel from the denominator, so a step whose demonstration carries no label text is scored on the channels that are actually observable. The weights follow prior multi-signal matchers, which treat recognized text as the primary evidence and recorded position as a weak tie-breaker rather than a locator in its own right (Nass et al., 2023; Yu et al., 2021); we set them once at specification time and never re-tuned them per scene, per perturbation class, or per benchmark. §5 reports all five fixed rules. The weights are global and frozen before evaluation; learning replaces the assumption that this one mixture should dominate every scene.

4 Experimental Setup

Training Corpus and Model Fitting Training set contains 38 designed web workflows spanning short single-page tasks, long sessions, and multipage careers and service flows. Each workflow has one clean demonstration and a deterministic terminal predicate. The demonstrations contain 504 actions in total (283 clicks, 115 selections, and 106 typed entries), with lengths from 5 to 35 steps.

For training, every workflow is rendered under 1 identity configuration and 9 perturbation classes (*Font size, viewport, theme, reorder, distractor, rename, Popup, Fake UI, new filed*) at three severities and 5 deterministic seeds in total of 5,168 rendered configurations. At each demonstrated step, RapidOCR, multi-scale template matching, and relative geometry generate the same candidates and 82 features used at inference. A private target pin labels correct candidates, but is excluded from screenshots, OCR, candidates, and model inputs. An oracle executes the demonstrated action only during supervision collection so that later training steps remain reachable. The resulting corpus contains 61,231 scenes, of which 59,992 contain at least one positive candidate and can supervise the ranker. The final model is fit on all training scenes and frozen before evaluation. Appendix B gives the roster, the training-side perturbation coverage, and two worked collection examples (Table 5 4 6).

Evaluation Benchmarks Test-set contained 46 workflows from MiniWoB++ that provides compact browser tasks with programmatic rewards (Liu et al., 2018) and human demonstrations released

with World of Bits (Shi et al., 2017). 179 workflows from WebArena provides self-hosted forum, commerce, administration, and software-development replicas with native functional evaluators (Zhou et al., 2024); we use the corresponding WONDERBREAD recordings (Wornow et al., 2024). 239 fully correct workflows from WebShop provides longer shopping trajectories, graded native reward, and released human sessions (Yao et al., 2022). After task conversion and clean-replay admission, each workflow is rendered under 1 identity configuration and 6 perturbation classes (*Font size, viewport, theme, reorder, distractor, rename*). Cell-level gating removes a perturbation only when it produces no measurable visual change, breaks the demonstrated task, makes a target unreachable, or invalidates the benchmark's native evaluator. The seven-condition main evaluation contains 285 MiniWoB++, 1,046 WebArena, and 1,339 WebShop task–perturbation cells per method, or 2,670 cells and 16,020 episodes across six methods. WebShop font size admits only 34 comparable cells and is marked exploratory wherever reported. Full condition is defined further in Appendix C.

Execution, Metrics, and Inference We compare 6 perturbations over other related works solutions that been summarized from Table 1: **Exact** reuses the recorded coordinate; **Text** Coppola et al. (2025) ranks live OCR boxes by label similarity; **Template** Yeh et al. (2009); Alégroth et al. (2015); Stocco et al. (2018) selects the strongest multi-scale match to the demonstrated target crop; **Geometry** Nass et al. (2023) ranks candidates by relative position; **Hybrid** Sereshkeh et al. (2020); Yu et al. (2021) combines text, template, and geometry using fixed global weights; and **Learned** applies our calibrated 82-feature resolver, with Hybrid retained as a confidence-dependent fallback. All methods receive the same demonstration, screenshot stream, OCR provider, action contract, and native evaluator. Only Learned uses fitted parameters with the training set. During evaluation, every cell is one free-running episode with no rollback, teacher forcing, nor oracle correction. We use each benchmark's native terminal evaluator for the cell score y_{wc}: MiniWoB++ and WebArena return binary success, while WebShop provides partial reward, which we retain and scale to 0–100. We compare methods separately for each benchmark on their shared cell roster, using workflow-level paired effects, 10,000-draw bootstrap intervals, and 10,000-draw paired sign-flip tests; WebArena bootstrap samples sites before workflows to account for within-site dependence.

From cells to a benchmark score. Because the admitted cell set C_w varies across workflows, we first compute $\bar{y}_w = |C_w|^{-1} \sum_{c \in C_w} y_{wc}$ and then report $\hat{\mu} = W^{-1} \sum_{w=1}^{W} \bar{y}_w$. This workflow-balanced aggregation gives every workflow equal weight, rather than weighting it by the number of cells that pass the admission gate.

Step-level diagnostics. With oracle target box $b_t^\star$, a step hit is $Y_t = \mathbf{1}[\mathrm{center}(b_{t\hat{\imath}_t}) \in b_t^\star]$; Appendix Figure 5 reports the workflow-balanced mean of prefix survival $S(k) = \prod_{t=1}^{k} Y_t$, alongside the i.i.d. reference $\bar{p}^N$, where $\bar{p}$ is the pooled per-step hit rate. These step-level diagnostics are reported separately from native episode outcomes.

5 Results and Discussion

Overall performance Our learned resolver is the strongest strategy on all three benchmarks, at 88.4% on MiniWoB++, 83.9% on WebArena, and 97.6 reward points on WebShop (Figure 3). Exact bootstrap intervals and paired-test results are reported in Appendix Table 10. It significantly outperforms every fixed strategy under the paired sign-flip test on MiniWoB++ and WebArena, and every geometry- or template-dependent strategy on WebShop.

Perturbations, rather than benchmark difficulty, are what break recorded replay. Coordinate replay is nearly exact on unperturbed identity cells, reaching 100.0% on MiniWoB++, 98.9% on WebArena, and 97.5% on WebShop (Appendix Table 11), but falls to 80.1%, 46.5%, and 65.2%, respectively, once the six perturbation classes are applied. In contrast, our resolver retains 88.4%, 83.9%, and 97.6% from identity scores of 97.8%, 95.5%, and 98.3%, indicating that the difference lies in robustness to perturbations rather than the ability to execute a demonstration. The per-class results explain why: each fixed strategy succeeds when its preferred evidence remains valid and fails when that evidence is disrupted. On WebArena, for example, coordinate replay scores 100.0% under theme and rename but only 1.2% under viewport and 9.6% under reorder; template matching reaches 79.3% under viewport but 0.0% under theme; and text matching falls to 31.2% under rename. Hybrid softens but does not eliminate these failures because a single global weighting cannot adapt when one channel becomes unreliable. Across the 18 benchmark–perturbation combinations, every fixed

strategy falls below 50% at least three times, and four of the five fall below 25% at least once, whereas our resolver's worst result is 71.4%. WebShop illustrates the boundary of this advantage: its controls retain stable labels, allowing Text (96.4%) and Hybrid (96.3%) to nearly match our resolver (97.6%), with no statistically significant difference ($p = 0.135$ and $p = 0.078$). Thus, the benefit of learned resolution is largest when a perturbation changes which evidence can be trusted; when one channel is already sufficient, as in WebShop, candidate-based strategies perform similarly.

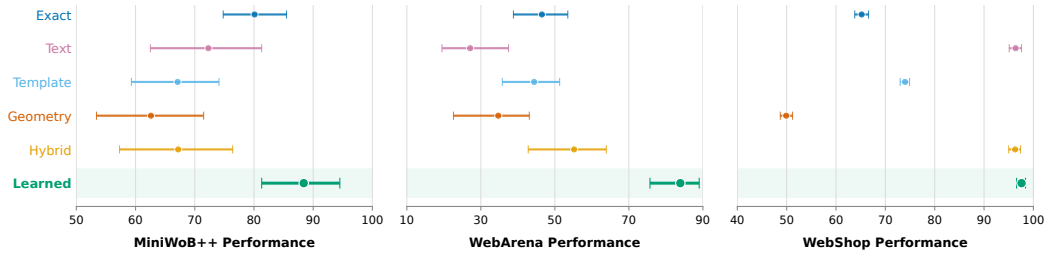


Figure 3: Average result across 6 type of perturbations compared with others methods.

Long-horizon performance A recorded workflow only pays off if it survives to its last step, so we next ask how far each strategy's robustness carries as demonstrations grow longer. Table 2 groups the admitted workflows by their number of recorded actions and scores each band workflow-balanced.

Table 2: Task length analysis. Workflows are grouped by their number of recorded actions and scored, workflow-balanced, over the 6 perturbation classes. W/C is workflows and paired cells per strategy; the bracket is a 95% bootstrap interval on our score and p is the paired sign-flip test of ours against Hybrid, the strongest fixed baseline.

Benchmark	Actions	W/C	Exact	Tex.	Temp.	Geo.	Hyb.	Learned [95% CI]	p
MiniWoB++	1–3	38/196	83.4	76.8	68.4	69.4	74.6	**93.0** [87.0, 97.8]	<0.0001
	4–6	6/31	69.4	56.9	61.9	29.4	31.4	**72.2** [44.4, 94.4]	0.0316
	7–15	2/12	50.0	33.3	**58.3**	33.3	33.3	50.0 [16.7, 83.3]	1.0000
WebArena	1–3	34/189	58.9	44.0	48.5	38.2	65.5	**86.5** [79.0, 98.0]	<0.0001
	4–6	37/185	45.1	49.0	46.9	35.4	64.5	**87.3** [76.6, 95.4]	<0.0001
	7–15	92/457	41.4	13.9	43.4	34.1	50.7	**83.6** [68.4, 90.4]	<0.0001
	≥16	7/36	**59.3**	4.8	25.0	21.4	16.7	57.1 [16.7, 94.4]	0.1221
WebShop	4–6	118/577	61.2	**97.3**	76.0	49.5	96.5	96.1 [94.4, 97.7]	0.7654
	7–15	87/385	66.8	96.7	73.8	48.7	98.0	**99.4** [98.9, 99.9]	0.0397
	≥16	34/134	74.9	92.5	67.6	54.5	91.1	**97.8** [95.8, 99.4]	0.0076

Length hurts fixed strategies but has little effect on ours. On WebArena, our score changes by less than three points from the shortest band to 7–15 actions, while Hybrid loses about one quarter of its score and Text about two thirds; our advantage over Hybrid therefore widens and remains significant. MiniWoB++ has too few long workflows for a meaningful comparison, while WebShop remains near ceiling because stable labels keep all candidate-based strategies above 90 (§5). The only band we do not lead, WebArena ≥16 actions, contains just seven workflows and no statistically separates.

Appendix Figure 5 examines the 99 WebArena workflows with at least seven actions. Our resolver preserves about three quarters through the shared seven-step prefix, while every fixed strategy falls below half. Yet all methods achieve higher end-to-end success than predicted from pooled per-step accuracy under independent errors, showing that failures cluster in a minority of hard workflow–perturbation cells (Appendix Table 12). Per-step accuracy alone therefore does not establish long-horizon reliability, motivating our use of native end-to-end success.

Unseen perturbations To test transfer to unseen perturbations, we refit the ranker 6 times, each time omitting one class, and evaluate it on that held-out class across all three benchmarks. Each fold includes identity controls to separate the effect of missing a class from that of using a smaller training

corpus; the difference-in-differences (DiD) subtracts this control loss. We count a loss only when its bootstrap interval excludes zero and its Holm-adjusted $p < 0.05$ (Table 3).

Table 3: Leave-one-perturbation-out transfer. *Trained* and *Unseen*: workflow-balanced scores of the fully trained and the refitted resolver on the held-out class. *Loss*: their paired difference [95% bootstrap CI]. *Identity*: the same difference on unperturbed control cells. *DiD*: Loss − Identity. p: paired sign-flip test, Holm-adjusted within benchmark. † marks a loss significant on both criteria.

Benchmark	Held-out class	W/C	Trained	Unseen	Loss [95% CI]	Identity	DiD	p
MiniWoB++	Font size	43/43	83.7	76.7	−7.0 [−18.6, 4.7]	+0.0	−7.0	0.4496
	Viewport	46/46	91.3	89.1	−2.2 [−10.9, 4.3]	+0.0	−2.2	1.0000
	Theme	42/42	90.5	71.4	−19.0 [−31.0, −7.1]	−2.2	−16.9	0.0084
	Reorder	42/42	85.7	76.2	−9.5 [−21.4, 0.0]	−4.3	−5.2	0.2214
	Distractor	44/44	88.6	86.4	−2.3 [−6.8, 0.0]	+0.0	−2.3	1.0000
	Rename	22/22	90.9	90.9	+0.0 [0.0, 0.0]	+0.0	+0.0	1.0000
WebArena	Font size	139/139	88.5	58.3	−30.2 [−51.7, −3.2]†	+0.6	−30.8	<0.0001
	Viewport	169/169	90.5	85.8	−4.7 [−10.6, 3.1]	−0.6	−4.2	0.0567
	Theme	166/166	78.3	54.8	−23.5 [−37.4, −8.7]†	−0.6	−22.9	<0.0001
	Reorder	94/94	83.0	61.7	−21.3 [−42.6, −4.4]†	−1.1	−20.2	<0.0001
	Distractor	161/161	71.4	64.6	−6.8 [−21.8, 5.8]	−0.6	−6.3	0.0132
	Rename	138/138	97.1	87.7	−9.4 [−20.2, 0.0]	−1.1	−8.3	0.0003
WebShop	Font size	34/34	97.1	91.2	−5.9 [−14.7, 0.0]	−1.7	−4.2	0.4997
	Viewport	239/239	99.2	94.6	−4.6 [−7.5, −2.1]†	−7.7	+3.1	0.0027
	Theme	239/239	99.4	93.3	−6.1 [−9.5, −3.1]†	−9.3	+3.2	0.0003
	Reorder	110/110	92.2	88.9	−3.3 [−9.3, 2.7]	−7.0	+3.7	0.3092
	Distractor	239/239	96.7	82.9	−13.8 [−19.0, −8.7]†	−6.3	−7.5	<0.0001
	Rename	239/239	97.1	90.8	−6.3 [−10.5, −2.5]†	−9.2	+2.9	0.0043

Exposure matters most when a perturbation removes a cue entirely. On WebArena, omitting font size, theme, or reorder costs 21–30 points while identity controls barely change; MiniWoB++ shows the same pattern for theme. These classes invalidate the recorded crop, geometry, or both, and also collapse fixed channels (Appendix Table 11), so the ranker must learn which remaining cue to trust.

The remaining rows of Table 3 are the classes that survive removal, and they are the ones whose evidence another represented class already supplies. Viewport is the clearest case, losing little on WebArena with an interval that crosses zero, which is what one expects when the other spatial perturbations in training already teach the geometry cues it needs. distractor and rename on WebArena have small sign-flip p values but intervals that include zero, and we do not count them. WebShop needs its control column to be read before its losses are interpreted: for 4/6 classes the identity cells drop *further* than the held-out class does, giving positive difference-in-differences, so those apparent losses are the smaller corpus rather than the missing perturbation. Distractor is the one WebShop class with a genuine class-specific loss, and the only counted loss anywhere that comes from an added element rather than a modified one.

The conclusion is a boundary on the claim rather than a caveat about it. The frozen resolver transfers to unseen websites, unseen layouts, and unseen tasks within a represented perturbation vocabulary. It does not learn a universally perturbation-invariant notion of a target. Extending the vocabulary is therefore a data question, and the classes worth adding first are those that eliminate a channel rather than merely perturb it.

Ablation: structured text versus pixels To test whether our gains depend on OCR, we run a paired ablation on WebArena GitLab, whose accessible element tree provides an alternative text channel. Each of its 208 admitted cells (43 workflows across six perturbation classes) is replayed with either OCR boxes or visible DOM-text boxes clipped to the same frame. The trace, perturbation, seed, and admitted roster are fixed; only the text channel changes.

The text channel does not materially affect aggregate performance (Figure 4): no interval excludes zero, every Holm-adjusted p is 1, and the strategy ranking is unchanged. Our resolver scores 86.9 with OCR and 86.0 with DOM text, retaining its margin over Hybrid. Its gains therefore neither require DOM access nor arise from OCR quality on this site. The only significant class-level change is Hybrid on *Reorder*, where DOM text improves success by more than 50 points (Appendix Table 13).

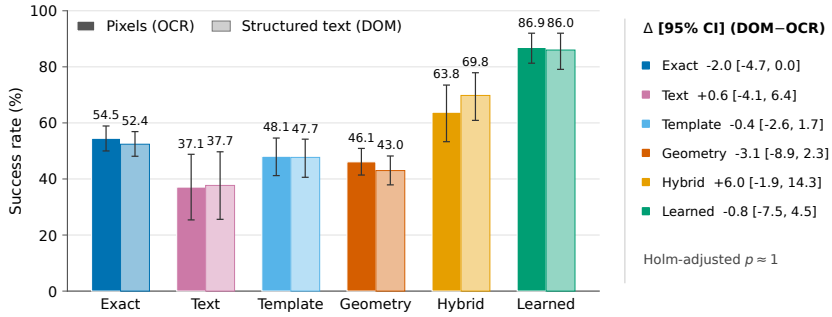


Figure 4: Perception-channel ablation on the WebArena GitLab site. Bars = 95% confidence intervals.

Fixed text–geometry weights cannot distinguish a moved target from a mismatch; our resolver already scores in the mid-eighties from pixels and shows no significant channel effect. This is consistent with its perturbation-class robustness in §5.

Further discussion The harness is designed for high-volume replay after a valid workflow has been demonstrated, such as repeated form filling, email or ticket handling, and administrative updates. A trace may be recorded by a person or produced by another system, after which our method executes it using a bounded candidate set, a fixed action vocabulary, and fail-closed recovery. Each action remains traceable to the recorded step, candidate scores, confidence, and any repair event. The method therefore preserves the efficiency and auditability of replay, but is not intended for unconstrained goal interpretation, planning, or generation.

As discussed in §1, we focus only on the *reference* problem. Our method is therefore not directly comparable to computer-use agents that must also infer the procedure. We instead compare the FLOPs required per replayed step, using the standard $2ND$ estimate for transformer calls and analytic counts for classical operations. Hybrid and learned re-grounding both require $\sim 10^{10}$ FLOPs, dominated by their shared perception pipeline, while our resolver adds only $\sim 10^{4}$ operations. In contrast, methods that invoke generative models require $\sim 10^{13}$–10^{15} FLOPs per step. Our robustness gains therefore come at negligible marginal compute over Hybrid. Estimates are provided in Appendix F Table 14.

6 Conclusion and Limitations

This work makes a narrow but common automation contract more reliable: given one complete demonstration, preserve its procedure and re-ground its recorded actions after the interface changes. A calibrated candidate ranker learns how the reliability of OCR, template, and geometry evidence varies from scene to scene while retaining Hybrid as a fallback. The model is trained only on 38 designed synthetic workflows and evaluated, without DOM access or benchmark-side tuning, on MiniWoB++, WebArena, and WebShop under identity and 6 shared perturbation classes. Learned arbitration produces large gains on MiniWoB++ and WebArena and near-parity with strong fixed methods on WebShop. It should therefore be viewed as an efficient execution harness for a human- or system-supplied trace, not as a planner or a general computer-use agent.

Our evaluation covers only common perturbations, not the full range of real-world variation. We study reliable reference grounding given a correct procedure; our method does not replan or recover from failures and is not a general-purpose agent. In this work, we only focus on repetitive daily tasks could be automated and use web-based benchmarks to enable controlled perturbations. Its reliability on longer workflows and native OS or application interfaces therefore remains untested.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript organization, language editing, code review, and selected implementation support. All AI-assisted research content, mathematical statements, experimental descriptions, code, and manuscript text were reviewed by the authors. Empirical claims are based on the project data and saved experiment outputs rather than on model-generated values. The authors take responsibility for the final content of the work, including text, claims, code, and artifacts produced with the aid of generative AI.

REFERENCES

Emil Alégroth, Robert Feldt, and Lisa Ryrholm. Visual GUI testing in practice: Challenges, problems and limitations. *Empirical Software Engineering*, 20(3):694–744, 2015.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. SeeClick: Harnessing GUI grounding for advanced visual GUI agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9313–9332, Bangkok, Thailand, 2024. Association for Computational Linguistics.

Shauvik Roy Choudhary, Dan Zhang, and Alessandro Orso. WATER: Web application TEst repair. In *Proceedings of the International Workshop on End-to-End Test Script Engineering (ETSE)*, pp. 24–29, 2011.

Riccardo Coppola, Robert Feldt, Michel Nass, and Emil Alégroth. Ranking approaches for similarity-based web element location. *Journal of Systems and Software*, 222:112286, 2025.

Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the digital world as humans do: Universal visual grounding for GUI agents, 2024. URL https://arxiv.org/abs/2410.05243. Published at ICLR 2025 (Oral).

Mouna Hammoudi, Gregg Rothermel, and Paolo Tonella. Why do record/replay tests of web applications break? In *Proceedings of the IEEE International Conference on Software Testing, Verification and Validation (ICST)*, pp. 180–190. IEEE, 2016.

Maurizio Leotta, Diego Clerissi, Filippo Ricca, and Cristiano Spadaro. Improving test suites maintainability with the page object pattern: An industrial case study. In *Proceedings of the IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pp. 108–113. IEEE, 2013.

Maurizio Leotta, Andrea Stocco, Filippo Ricca, and Paolo Tonella. Using multi-locators to increase the robustness of web test cases. In *Proceedings of the IEEE International Conference on Software Testing, Verification and Validation (ICST)*, pp. 1–10. IEEE, 2015.

Maurizio Leotta, Andrea Stocco, Filippo Ricca, and Paolo Tonella. ROBULA+: An algorithm for generating robust XPath locators for web testing. *Journal of Software: Evolution and Process*, 28 (3):177–204, 2016.

Gilly Leshed, Eben M. Haber, Tara Matthews, and Tessa Lau. CoScripter: Automating & sharing how-to knowledge in the enterprise. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pp. 1719–1728. ACM, 2008.

Evan Zheran Liu, Kelvin Guu, Panupat Pasupat, and Tianlin Shi. Reinforcement learning on web interfaces using workflow-guided exploration. In *International Conference on Learning Representations*, 2018.

Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. OmniParser for pure vision based GUI agent, 2024. URL https://arxiv.org/abs/2408.00203.

Michel Nass, Emil Alégroth, Robert Feldt, Maurizio Leotta, and Filippo Ricca. Similarity-based web element localization for robust test automation. *ACM Transactions on Software Engineering and Methodology*, 32(3):1–30, 2023.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. UI-TARS: Pioneering automated GUI interaction with native agents, 2025. URL https://arxiv.org/abs/2501.12326.

Alborz Rezazadeh Sereshkeh, Gary Leung, Krish Perumal, Caleb Phillips, Minfan Zhang, Afsaneh Fazly, and Iqbal Mohomed. VASTA: A vision and language-assisted smartphone task automation system. In *Proceedings of the 25th International Conference on Intelligent User Interfaces (IUI)*, pp. 22–32, 2020.

Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pp. 3135–3144. PMLR, 2017.

Andrea Stocco, Rahulkrishna Yandrapally, and Ali Mesbah. Visual web test repair. In *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, pp. 503–514, 2018.

Wil M. P. van der Aalst, Martin Bichler, and Armin Heinzl. Robotic process automation. *Business & Information Systems Engineering*, 60(4):269–272, 2018.

Yangyue Wang, Harshvardhan Sikka, Yash Mathur, Tony Zhou, Jinu Nyachhyon, and Pranav Guruprasad. GUI-Perturbed: Domain randomization reveals systematic brittleness in GUI grounding models, 2026. URL https://arxiv.org/abs/2604.14262.

Michael Wornow, Avanika Narayan, Krista Opsahl-Ong, Quinn McIntyre, and Nigam H. Shah. WONDERBREAD: A benchmark for evaluating multimodal foundation models on business process management tasks. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, pp. 20744–20757, 2022.

Tom Yeh, Tsung-Hsiang Chang, and Robert C. Miller. Sikuli: Using GUI screenshots for search and automation. In *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology (UIST)*, pp. 183–192. ACM, 2009.

Shengcheng Yu, Chunrong Fang, Yexiao Yun, and Yang Feng. Layout and image recognition driving cross-platform automated mobile testing. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pp. 1561–1571. IEEE, 2021.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024.

A Perturbation Details

All perturbations are deterministic given workflow, class, severity, and seed. Font size changes typography scale and natural wrapping; viewport changes browser dimensions and responsive layout; theme changes palette, typography, borders, and control styling; reorder permutes eligible sibling groups; distractor inserts plausible decoy controls or overlays; and rename substitutes semantically related visible labels. These six classes appear in both training and the main evaluation. Training additionally includes popup, fake-UI, and new-field classes. Spacing changes margins, padding, and line height, is absent from training, and is reserved for a separate ablation rather than pooled into the main result. Table 4 summarizes the perturbation coverage and rendering settings used in training and evaluation.

Table 4: Perturbation coverage and rendering settings for training and evaluation.

Perturbations	Train (38)	Benchmark (464)
Identity	✓	✓
Font size, viewport, theme, reorder, distractor, rename	✓	✓
Popup, fake UI, new field	✓	✗
Spacing	✗	✓
# Perturbations	**9**	**6**
Severities × seeds	3×5	one frozen setting

B The Training Set Corpus

The training set contained 38 workflows cover twelve task families across the roster the reference demonstrations total 504 steps (5 to 35 per workflow, median 9), composed of 283 clicks, 115 selections, and 106 typed entries. It contains six short and long single-page interaction styles, and two genuine multipage systems. Table 5 reports the full composition. Workflows within a row share an application theme or task domain, but every workflow has its own clean demonstration and terminal predicate.

Table 5: The 38 OOD38 workflow units *Pg.* counts distinct HTML documents; single-document applications switch views in place. *Steps* is the length of the unit's one reference demonstration, decomposed into clicks (Cl), native `select` choices (Se), and typed entries (Ty).

Unit	Pg.	Steps	Cl	Se	Ty	Task
Admin console						
`admin`	1	5	2	2	1	Filter the user table, edit a row's role and status, save
`admin_create_user`	1	9	5	2	2	Create a user: name, email, role, status, invite, confirm
Appointment booking						
`booking_cancel`	1	7	6	1	0	Open a booking, cancel it with a reason, confirm in dialog
`booking_resched`	1	8	8	0	0	Reschedule a booking to a new date and slot, confirm
`booking_slot`	1	9	5	1	3	Book a service: pick date and slot, fill contact details
Careers portal (multipage)						
`career_alert_and_application`	7	21	12	6	3	Create a job alert, then apply to the alerted role
`career_apply_full`	9	31	13	8	10	Search, open a role, create an account, apply, review, submit
`career_apply_hard`	10	35	14	11	10	As above plus eligibility and EEO pages, longest workflow
`career_apply_saved_profile`	6	21	12	4	5	Sign in, reuse a saved profile, update it, apply

Table 5 (continued)

Unit	Pg.	Steps	Cl	Se	Ty	Task
CRM						
crm_call	1	8	5	1	2	Log a call on a contact: duration, notes, outcome, follow-up
crm_lead	1	8	4	1	3	Create a lead: company, contact, source, priority, follow-up
crm_stage	1	7	3	2	2	Advance a deal stage, set amount and owner, mark won
Document management						
docs_move	1	7	6	0	1	Move a file: search the tree, expand a folder, keep a copy
docs_rename	1	7	5	1	1	Rename a file via the overflow menu, set extension, keep history
docs_share	1	8	5	1	2	Share a file: add a recipient, set permission, notify, message
Email						
email	1	5	2	0	3	Compose and send a message
email_reply_archive	1	9	7	0	2	Filter by sender, reply to a thread, archive it
Email (long)						
email_long	1	22	16	1	5	Morning triage: reply, archive, forward, label across threads
Job application						
jobapp	1	6	3	1	2	Open a posting and submit an application form
jobapp_alert	1	9	5	3	1	Filter postings, save a job, then create a job alert
Job application (long A)						
jobapp_long	1	28	16	8	4	Repeated search-apply sessions across several categories
Job application (long B)						
jobapp_long_b	1	15	7	5	3	Filter, apply to a posting, then configure an alert
IT service desk (multipage)						
service_access_request	6	21	9	6	6	Request warehouse-system access: requestor, role, review
service_change_hard	7	31	10	11	10	File a database change request with impact and review pages
service_equipment_order	6	22	11	6	5	Configure and order a laptop with delivery details
service_incident_escalation	6	22	12	6	4	Escalate an incident with impact assessment and review
Account settings						
settings	1	5	4	1	0	Appearance tab: dark mode, compact mode, accent colour, save
settings_notifications	1	9	6	3	0	Notification tab: channels, sound, digest, quiet hours
Account settings (long)						
settings_long	1	20	14	4	2	Profile, security, and notification panes saved in sequence
Shopping						
shopping	1	5	4	0	1	Search a product, add to cart, check out
shopping_coupon	1	10	5	4	1	Filter and sort, set quantity, apply a coupon, check out
Shopping (long)						
shopping_long	1	16	10	4	2	Add two products from different categories, coupon, check out
Help desk						

Table 5 (continued)

Unit	Pg.	Steps	Cl	Se	Ty	Task
`support_kb`	1	7	6	0	1	Search the knowledge base, rate an article, open a related one
`support_ticket`	1	9	4	3	2	File a ticket: product, category, severity, description, consent
`support_triage`	1	9	5	3	1	Triage a queued ticket: priority, assignee, status, note
Help desk (long)						
`support_long`	1	18	10	4	4	Knowledge-base search, file a ticket, then triage the queue
Surveys						
`survey`	1	5	5	0	0	Answer a two-step customer survey and submit
`survey_event_reg`	1	10	7	1	2	Register for an event: sessions, contact details, meal choice
Total (38 units)		**504**	**283**	**115**	**106**	

The collection matrix contains one identity configuration plus nine perturbation classes at three severities and five seeds, giving 136 configurations per workflow and 5,168 configurations in total. Candidate supervision is collected from screenshots using RapidOCR, template peaks, and relative geometry. A private target pin supplies only the binary candidate label. Runtime assertions prohibit that pin, DOM selectors, application identity, perturbation identity, severity, seed, and filenames from entering the feature vector.

The collection contains 61,231 step-level scenes and 1,616,022 candidate rows, including 121,931 positive rows. The 1,239 scenes with no positive candidate are retained for candidate-recall accounting but cannot supervise a ranker, leaving 59,992 usable scenes. Model fitting, calibration, and hyperparameter selection use only this corpus. The three benchmark datasets remain unavailable until the complete training artifact is frozen.

B.1 How a Supervised Scene Is Collected: Two Worked Examples

Collection procedure. A configuration is one $(\text{unit}, \text{class}, \text{severity}, \text{seed})$ tuple. The collector opens the unit at the perturbed URL in a headless browser—at the demonstration geometry of 1280×800, or at the perturbed dimensions when the class is viewport—and then, for each demonstrated step in order: (i) restores the scroll position recorded in the demonstration; (ii) captures the screenshot; (iii) extracts text boxes through the same channel the replay strategies see at inference; (iv) loads the demonstration crop for that step as the template; (v) builds the candidate set and the 82-dimensional feature vector with the identical module used at inference; (vi) queries the harness-only target pin for the *live* ground-truth box and labels a candidate positive when its center falls inside that box; (vii) emits one row per candidate carrying provenance $(\text{unit}, \text{class}, \text{severity}, \text{seed}, \text{step}, \text{candidate})$, the label, and the features; and (viii) advances by executing the oracle action on the pinned element, so a ranking error at one step never censors supervision for later steps. The pin enters only stage (vi) and (viii); it is absent from the screenshot, the text boxes, the candidate generator, and every feature, and a runtime whitelist assertion over the feature names enforces that separation. The two examples below are real scenes drawn from the frozen collection; both are cases Hybrid ranks incorrectly, which is precisely the supervision a learned ranker needs.

Example 1: text and template both fail, geometry decides. Unit `jobapp`, class rename, severity 3, seed 1, step 1. The demonstration clicks the second `Apply` button on a careers listing; the recorded label text is `Apply` and two sibling postings carry visually identical `Apply` buttons. The rename perturbation rewrites the visible label of the target control while leaving its siblings untouched. The scene yields 23 candidates, three of which overlap the live pin box. As the upper panel of Table 6 shows, the true target's OCR text similarity collapses to 0.26 and its global template correlation to 0.00, while both unrenamed siblings retain 1.00 on each. The Hybrid therefore ranks the two wrong buttons first and second and the true target third. The only channel that still separates them is relative geometry: the target sits essentially where the demonstration left it (0.997, radial displacement 0.020),

whereas the siblings are displaced by roughly an eighth of the viewport diagonal. Learning to prefer geometry *when and only when* text and template have jointly degraded is what this row teaches.

Example 2: geometry fails, text and neighbourhood decide. Unit `support_triage`, class reorder, severity 3, seed 3, step 3. The demonstration sets a queued ticket's priority dropdown—reading `Medium` in the demonstration—to `High`. Reordering the queue rows moves the selected ticket, and with it the detail panel, far from where the demonstration recorded it. The scene yields 43 candidates, two of which overlap the live pin box. The lower panel of Table 6 is the mirror image of Example 1: the target's text similarity is a perfect 1.000, but its geometry prior collapses to 0.105 at a radial displacement of 0.531 of the viewport diagonal, while two *other* dropdowns on the page also read `Medium` with text similarity 1.000 and far better geometry. Text alone, geometry alone, and Hybrid all place the true target third. What separates it here is local context: the demonstrated neighbours (`PRIORITY`, `ASSIGNEE`, `INTERNAL NOTE`) match the target's neighbourhood exactly (1.000) and the decoys' only weakly (0.514)—the same neighbourhood channel that was uninformative in Example 1, where every candidate scored 1.000.

Table 6: Two collected scenes, ranked by the Hybrid score. $*$ marks a candidate whose center lies inside the live target box (label $y = 1$). Columns are feature values as recorded in the corpus: demonstrated-versus-live text similarity, global template correlation, the relative-geometry prior, radial displacement from the demonstrated location in units of the viewport diagonal, best neighbourhood-text similarity, and the Hybrid score.

Rank	Candidate	Tex.	Temp.	Geo.	Radial	Neigh.	Hyb.
Example 1 — `jobapp`, rename, severity 3, seed 1, step 1 (23 candidates, 3 positive)							
1	sibling `Apply`	1.000	1.000	0.892	0.120	1.000	0.894
2	sibling `Apply`	1.000	1.000	0.881	0.126	1.000	0.892
3	**renamed target***	0.257	0.000	0.997	0.020	1.000	0.530
4	target, template peak*	0.000	0.821	0.986	0.041	1.000	0.485
5	target, demo projection*	0.000	0.707	1.000	0.003	1.000	0.448
Example 2 — `support_triage`, reorder, severity 3, seed 3, step 3 (43 candidates, 2 positive)							
1	decoy `Medium`	1.000	0.000	0.680	0.219	0.514	0.648
2	decoy `Medium`	1.000	0.000	0.756	0.187	0.514	0.642
3	**moved target***	1.000	0.000	0.105	0.531	1.000	0.542
4	target, template peak*	0.000	0.912	0.214	0.439	1.000	0.362
5	unrelated control	0.450	0.000	0.779	0.177	0.778	0.358

Why both are needed. The two examples require opposite resolutions from the same feature vector: Example 1 demands that geometry override text and template, Example 2 demands that text and neighbourhood override geometry. No single fixed weighting over the three channels ranks both correctly, which is the concrete form of the argument made in §3, and the reason supervision is collected per candidate rather than per channel. Enumerating the full grid rather than sampling it guarantees that both regimes appear at every severity and under every nuisance seed. In total the frozen collection contains 61,231 step-level scenes and 1,616,022 candidate rows, of which 121,931 rows are positive; the 1,239 scenes with no positive candidate are retained in candidate-recall accounting but cannot supervise a ranker, leaving 59,992 usable scenes.

C Benchmark Coverage and Admission

The benchmark roster is derived from public benchmark tasks and demonstrations. This section is an extension of §4: Table 7 traces every published task down to an executed episode, and the remainder of the section states the four gates that produce those reductions, together with the environment handling and the audit record. Every reduction is an enumerated decision rather than a sample: rejected tasks and rejected cells are retained as explicit rows with reasons, which is why the table can report the rejected count on its own line.

Table 7: From published benchmark to executed episode. The three horizontal blocks use different denominators: 3,248 workflow–perturbation cells are enumerated and given an explicit disposition, 2,670 are executed per method, and 2,636 remain after the exploratory WebShop font-size cells are removed.

Stage	WebArena	MiniWoB++	WebShop	Total
Tasks in the published benchmark	812	104	500	1,416
Tasks with ≥ 1 public human demonstration	598	58	498	1,154
Human demonstration recordings available	2,928	17,971	1,643	22,542
In scope for demonstration replay	390	58	498	946
Demonstration compiles to the replay action space	377	47	476	900
Clean replay admitted by the native evaluator[†]	179	46	462	687
Human demonstration attains the full native reward[‡]	179	46	239	464
Enumerated workflow–perturbation cells	1,253	322	1,673	3,248
rejected by perturbation gating before execution	207	37	334	578
Episodes executed per method[§]	**1,046**	**285**	**1,339**	**2,670**
confirmatory (exploratory cells removed)	1,046	285	1,305	2,636

[†]For MiniWoB++, this row additionally requires a rematerialized trace loadable by the grid runner; the one compilable but unsupported `guess-number` workflow is therefore excluded from the 46-workflow roster. [‡]MiniWoB++ and WebArena score a binary native outcome, so their admitted demonstrations are already correct by construction and this gate removes nothing. WebShop grades its human sessions: of the 462 admitted workflows only 239 have a demonstration that earns the full reward, and the remaining 223 (208 partially rewarded, 15 scoring zero) are discarded so that no strategy is charged for a demonstrated error. [§]Exact, Text, Template, Geometry, Hybrid, and Learned run on the identical executed set, making the comparison paired. The 34 executed WebShop font-size cells are exploratory and are excluded from the confirmatory row.

The four gates. *Scope* removes tasks that the replay contract cannot express. WebArena's 208 free-text-answer tasks are scored by string match, whereas the replayer emits no final natural-language answer; this accounts for the reduction from 598 demonstrated tasks to the 390-workflow replay pool. MiniWoB++ instead loses 46 of its 104 published tasks because they have no eligible public human demonstration. *Conversion* requires every demonstrated step to compile into the common click/type/select action contract while preserving action order, typed values, and compatibility with the source evaluator. When several demonstrations exist for a task, the canonical trace is the one with median grounded-action count, avoiding both unusually short recordings and long traces containing annotator retries. This gate retains 377 WebArena, 47 MiniWoB++, and 476 WebShop workflows. *Admission* requires a teacher-forced rematerialization on a pristine instance and a free-running identity episode whose final state is accepted by the native evaluator. For WebShop, acceptance means reproducing the demonstration's recorded reward and that reward being the benchmark's full 1.0; the 223 admitted workflows whose human session earns less than full reward are discarded, so that on all three benchmarks the demonstration given to every strategy is a correct one. MiniWoB++ additionally requires a rematerialized trace loadable by the grid runner, which excludes the otherwise compilable `guess-number` workflow. Admission leaves 179 WebArena, 46 MiniWoB++, and 239 WebShop workflows, or 464 in total. *Cell gating* acts last on the perturbation grid rather than on the workflow roster. Crossing the 464 admitted workflows with identity and the six shared perturbation classes enumerates 3,248 workflow–perturbation cells. Gating rejects 578 cells whose perturbation is ineffective, violates task semantics, makes a target unobservable or unactionable, or fails the private-oracle check, leaving 2,670 episodes executed per method. After the 34 exploratory WebShop font-size cells are removed, 2,636 cells remain confirmatory.

Per-perturbation denominators. Cell gating does not act uniformly across perturbations, so the executed count for a perturbation can be substantially smaller than the admitted workflow roster. Table 8 reports every benchmark–perturbation denominator. Identity retains all 464 admitted workflows, whereas the perturbation classes retain between 216 font-size cells and 454 viewport cells. A cell is rejected when the perturbation produces no measurable visual change, compromises task semantics or evaluator validity, or moves a required target outside the observable and actionable region under the frozen replay protocol. This is especially consequential for WebShop: font-size changes retain only 34 of 239 workflows and are therefore treated as exploratory, while reorder

retains 110. Success rates are always computed over the executed count for the corresponding benchmark–perturbation pair, never over the full enumerated grid; the 34 exploratory WebShop font-size cells are also excluded from confirmatory aggregates.

Table 8: Executed task–perturbation cells per method. Parentheses show the admitted workflow roster. The WebShop font-size cell set is exploratory.

Benchmark	Identity	Font	Viewport	Theme	Reorder	Distr.	Rename	Total
MiniWoB++ (46)	46	43	46	42	42	44	22	285
WebArena (179)	179	139	169	166	94	161	138	1,046
WebShop (239)	239	*34*	239	239	110	239	239	1,339
Total	464	216	454	447	246	444	399	**2,670**

Per-cell admission. Benchmarks admission has a static and a dynamic stage. Static preflight requires a measurable perturbation, actionable targets, readable text when text should remain readable, unchanged task semantics, and a valid native evaluator. A private oracle then re-enacts the source demonstration on the perturbed page, and a cell is evaluated only if this execution satisfies the source task. The private target pin is available to admission and scoring but is excluded from screenshots, OCR, candidates, features, and model inputs.

Environment handling. WebArena demonstrations are re-enacted in a controlled Playwright environment at fixed capture geometry, which removes browser-chrome, scaling, and viewport inconsistencies from the recorded coordinates. WebArena restores its mutable containers before every actuating episode, WebShop uses isolated warm servers, and MiniWoB++ is stateless. WebArena spacing and WebShop font size are reported as exploratory, because their benchmark implementations do not satisfy the comparability rule on every site.

Audit record. The runner retains rejected cells with reasons. One unsupported MiniWoB++ `guess-number` entry contributes eight explicit static rejections. Five Magento-admin reorder cells are structural exclusions because restoration is not deterministic. Popup and new-field external variants were removed before the frozen wave because their converters did not preserve source-task validity.

D Model and Feature Details

The 82-feature whitelist groups signals into demonstrated/live text similarity, global and local template scores, relative geometry, size and aspect change, neighboring-text agreement, candidate-source indicators, within-scene ranks and margins, channel agreement, viewport normalization, and step context. Missing channels receive explicit availability indicators and do not silently become low similarity.

Table 9: Composition of the 82-dimensional candidate feature vector.

Feature group	Count
Signal scores (text: 4; template: 6; geometry and Hybrid: 2)	12
Local template NCC and peak support	6
Normalized geometry	12
Neighbor context	3
Availability and candidate-source flags	10
Relative rank, margin, and contrast	21
Broadcast scene context	18
Total	82

The histogram gradient-boosted ranker uses at most 15 leaves per tree, minimum leaf support 40, learning rate 0.05, $L_2 = 1$, 100 boosting iterations, disabled internal early stopping, and

seed 42. Positive candidate rows receive weight $\min(N_-/N_+, 9)$ and negative rows receive unit weight. Monotonic constraints require text similarity, template agreement, geometry prior, neighbor agreement, reciprocal ranks, and the Hybrid score to be non-decreasing; fractional rank features are non-increasing. The confidence calibrator receives maximum model probability and channel-agreement count. Calibration and blend parameters are selected solely from grouped out-of-fold predictions inside training-set-38; no outcome enters selection.

E Statistical Details

The primary statistic uses paired learned and Hybrid outcomes on the six WebArena perturbation classes. Cell differences are averaged within workflow so that a workflow with more admitted perturbations receives no additional weight. A two-sided randomization test independently flips each workflow difference in 10,000 seeded draws and uses a plus-one correction. The 95% interval uses 10,000 hierarchical bootstrap draws, sampling sites and then workflows within sampled sites.

MiniWoB++ and WebArena perturbation rows use the native binary task outcome. WebShop perturbation rows average available native rewards; infrastructure timeouts with no native reward remain explicitly recorded, and per-method reward denominators are provided in `generated/main_results.csv`. All descriptive micro-averages weight executed cells, not benchmark names or published task counts.

E.1 Perturbation performance breakdown over benchmarks

Table 10: Average result across 6 type of perturbations compared with others methods. Brackets give 95% bootstrap intervals; p is the paired sign-flip test of ours against that strategy.

	Prior resolution strategies					Ours
	Exact	Tex.	Temp.	Geo.	Hyb.	Learned
MiniWoB++ (46 workflows, 239 paired cells per strategy)						
Score	80.1	72.3	67.1	62.6	67.2	**88.4**
95% CI	[74.8, 85.5]	[62.5, 81.3]	[59.3, 74.1]	[53.4, 71.5]	[57.3, 76.4]	[81.3, 94.5]
p vs. ours	0.0351	0.0010	<0.0001	<0.0001	<0.0001	–
WebArena (170 workflows, 867 paired cells per strategy)						
Score	46.5	27.1	44.4	34.7	55.2	**83.9**
95% CI	[38.8, 53.5]	[19.5, 37.5]	[35.8, 51.3]	[22.6, 43.1]	[42.8, 63.9]	[75.7, 89.0]
p vs. ours	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001	–
WebShop (239 workflows, 1,096 paired cells per strategy)						
Score	65.2	96.4	74.0	49.9	96.3	**97.6**
95% CI	[63.8, 66.6]	[95.1, 97.6]	[73.0, 74.9]	[48.7, 51.2]	[95.0, 97.4]	[96.6, 98.4]
p vs. ours	<0.0001	0.1353	<0.0001	<0.0001	0.0779	–

E.2 Long-sequence step numbers

Table 12 lists the values plotted in Figure 5: prefix survival at every k in the shared seven-action window, the pooled per-step hit rate, and observed versus i.i.d.-predicted native success at full trace length, for all six strategies on the long external WebArena roster.

E.3 Perception ablation by perturbation class

Table 13 breaks the perception-channel ablation of §5 down by perturbation class for the two ranked strategies, Hybrid and our learned resolver. Cell counts vary by class because the oracle admits a class only where it produces a measurable, task-preserving change. Holm adjustment runs across all twelve class-level comparisons, and † marks the one difference that is significant on both criteria.

Table 11: pixel/OCR evaluation. MiniWoB++ and WebArena report native task success (%); WebShop reports mean native reward multiplied by 100. n is the admitted task–perturbation cells shared by all six methods. Bold marks the best value in each row. WebShop font size is exploratory because only the 34 cells satisfying the adapter's comparability gate are admitted.

Benchmark	Perturbation	n	Exact	Tex.	Temp.	Geo.	Hyb.	Learned
MiniWoB++	Identity	46	**100.0**	84.8	91.3	93.5	87.0	97.8
	Font size	43	74.4	67.4	65.1	74.4	72.1	**83.7**
	Viewport	46	**100.0**	84.8	89.1	56.5	80.4	91.3
	Theme	42	**100.0**	88.1	0.0	45.2	50.0	90.5
	Reorder	42	59.5	76.2	**85.7**	54.8	71.4	**85.7**
	Distractor	44	52.3	72.7	81.8	63.6	79.5	**88.6**
	Rename	22	**100.0**	13.6	90.9	95.5	27.3	90.9
WebArena	Identity	179	**98.9**	40.8	77.7	88.3	77.1	95.5
	Font size	139	15.8	20.1	41.7	15.8	74.1	**88.5**
	Viewport	169	1.2	24.9	79.3	7.7	69.8	**90.5**
	Theme	166	**100.0**	36.7	0.0	34.9	29.5	78.3
	Reorder	94	9.6	17.0	81.9	14.9	72.3	**83.0**
	Distractor	161	40.4	28.6	16.8	48.4	36.6	**71.4**
	Rename	138	**100.0**	31.2	65.9	86.2	68.1	97.1
WebShop	Identity	239	97.5	98.7	**100.0**	**100.0**	98.5	98.3
	Font size†	34	0.0	**100.0**	72.8	0.0	**100.0**	97.1
	Viewport	239	0.0	98.6	**100.0**	3.2	98.5	99.2
	Theme	239	98.3	98.6	0.0	6.3	97.6	**99.4**
	Reorder	110	0.0	**98.7**	93.2	36.8	95.7	92.2
	Distractor	239	97.1	92.6	88.8	**99.7**	91.4	96.7
	Rename	239	97.9	94.6	**100.0**	**100.0**	98.0	97.1

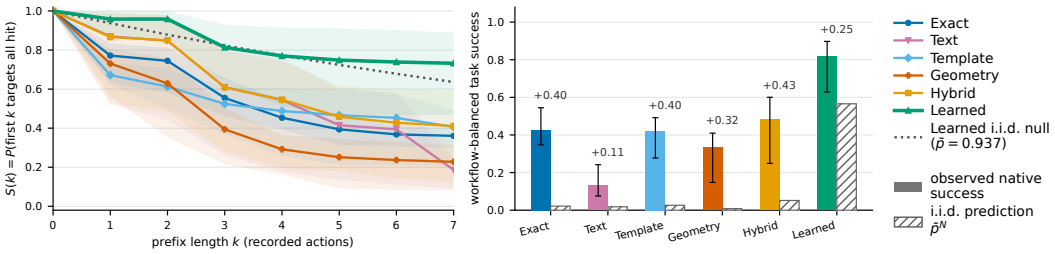


Figure 5: Step-level behaviour on the 99 WebArena workflows with at least seven recorded actions (493 paired cells over the six perturbation classes). **Left** Workflow-balanced prefix survival $S(k)$ over the seven-action window every workflow shares, with our method's i.i.d. reference $\bar{p}^{\,k}$ dotted. **Right** Native task success at each trace's full length against the $\bar{p}^{\,N}$ prediction under independent step hits, annotated with observed minus predicted. Bands and error bars are 95% site-then-workflow bootstrap intervals; exact values are in Appendix Table 12.

F Compute Accounting

Table 14 gives the per-step arithmetic estimates referenced in §5. Counts are device-independent: transformer forward passes use the standard $2ND$ estimate for N parameters and D tokens, and classical operations are counted analytically.

Table 12: Long-sequence step numbers plotted in Figure 5. $S(k)$ is workflow-balanced prefix survival on the 99 external WebArena workflows with at least 7 recorded actions (493 paired perturbation cells per method); $\bar{p}$ is the pooled per-step hit rate, *Native* the observed workflow-balanced task success at full trace length, and $\bar{p}^{\,N}$ its i.i.d. prediction. Brackets are 95% site-then-workflow bootstrap intervals.

	Prefix survival $S(k)$ (%)							Full-length outcome (%)		
Method	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$\bar{p}$	Native [95% CI]	$\bar{p}^{\,N}$
Exact	77.2	74.5	55.5	45.3	39.4	36.8	36.1	61.7	42.7 [34.8, 54.5]	2.2
Tex.	87.0	84.8	61.0	54.4	41.5	39.4	18.6	60.2	13.2 [7.5, 24.2]	1.8
Temp.	67.2	61.2	52.4	48.8	46.6	45.3	40.7	63.2	42.1 [27.8, 49.2]	2.6
Geo.	73.1	62.9	39.4	29.2	25.1	23.7	22.8	53.9	33.2 [14.8, 41.0]	0.8
Hyb.	86.6	85.0	60.9	54.6	45.9	42.8	41.1	69.2	48.2 [24.9, 60.0]	5.2
Learned	95.9	95.9	81.3	77.0	74.8	73.9	73.2	93.7	81.7 [62.8, 89.8]	56.6

Table 13: Perception-channel ablation by perturbation class on WebArena GitLab. *Pixels* and *DOM* are workflow-balanced success (%) within the class; Δ is the paired per-workflow DOM$-$pixel difference with a 95% bootstrap interval; p is the Holm-adjusted sign-flip test across the twelve comparisons.

Strategy	Perturbation class	n	Pixels	DOM	Δ [95% CI]	p
Hybrid	Font size	25	88.0	64.0	-24.0 [-40.0, -8.0]	0.3652
	Viewport	43	67.4	79.1	$+11.6$ [$+0.0$, $+23.3$]	1.0000
	Theme	43	53.5	48.8	-4.7 [-11.6, $+0.0$]	1.0000
	Reorder	19	21.1	73.7	$+52.6$ [$+31.6$, $+73.7$]†	0.0276
	Distractor	38	57.9	68.4	$+10.5$ [-7.9, $+28.9$]	1.0000
	Rename	40	82.5	87.5	$+5.0$ [-7.5, $+17.5$]	1.0000
Learned	Font size	25	96.0	96.0	$+0.0$ [$+0.0$, $+0.0$]	1.0000
	Viewport	43	95.3	88.4	-7.0 [-16.3, $+2.3$]	1.0000
	Theme	43	74.4	76.7	$+2.3$ [-7.0, $+14.0$]	1.0000
	Reorder	19	84.2	100.0	$+15.8$ [$+0.0$, $+31.6$]	1.0000
	Distractor	38	73.7	76.3	$+2.6$ [$+0.0$, $+7.9$]	1.0000
	Rename	40	100.0	95.0	-5.0 [-12.5, $+0.0$]	1.0000

Table 14: Estimated arithmetic per replayed step.

Approach	FLOPs/step	Device	Generative calls
Coordinate replay	$\sim 10^3$	local CPU	0
Hybrid	$\sim 10^{10}$	local CPU	0
Learned re-grounding (ours)	$\sim 10^{10}$	local CPU	0
Dedicated grounding VLM (7B)	$\sim 10^{13}$	cloud GPU	1
Frontier computer-use agent	10^{14}–10^{15}	cloud GPU cluster	≥ 1