

TraceFork: A System for Inspecting GUI-Agent Trajectory Forks

Yuzhang Zhong
University of Macau

Stephen Xia
Northwestern University

Abstract

Autonomous GUI and web agents frequently generate long, multi-step execution traces with rich environmental states, making manual inspection and fault localization challenging. Existing visualization tools typically render isolated single-run trajectories, obscuring fine-grained behavioral discrepancies across parallel runs on the same task. We introduce TRACEFORK, an interactive system designed to align, compare, and inspect paired same-task trajectories produced by AI agents or human demonstrators. TRACEFORK ingests execution traces from standardized benchmarks (e.g., WebArena, VisualWebArena) and custom user logs via an extensible normalization schema. Using a multi-modal dynamic programming alignment engine, the system collapses shared prefixes onto a single trunk, isolates behavioral divergence points into distinct branches, and identifies downstream reconvergence via conservative rejoin cues. Across a library of 1,722 tasks and 1,387 aligned trajectory pairs (33,337 steps), TRACEFORK achieves a median alignment latency of 45.5 ms per pair (p95 320.4 ms). On a manually annotated pilot subset, it achieves an F1 score of 0.910 for divergence localization, offering an open-source, deterministic diagnostic workbench for the agent research community.

1 Introduction

Autonomous web and GUI agents operate in complex environments by observing visual viewports, resolving target elements, typing text, and executing navigation actions. While recent benchmarks such as Mind2Web, WebArena, VisualWebArena, WorkArena, BrowserGym, WebLINX, and OSWorld (Deng et al., 2023; Zhou et al., 2024; Koh et al., 2024; Drouin et al., 2024; Le Sellier de Chezelles et al., 2024; Reddy et al., 2024; Xie et al., 2024) capture these interactions in granular detail, debugging agent behavior remains labor-intensive.

Most existing viewers display executions as linear logs or isolated replays one run at a time. When an agent fails or deviates, developers are forced to manually cross-reference parallel recordings to determine whether a minor misclick triggered a permanent failure or whether the agent successfully recovered downstream.

Comparing two trajectories side by side reveals essential diagnostic insights: Where did a model run first deviate from a reference human demonstration? Did an ambiguous visual distractor steer a vision-language model away from the intended button? Did an agent that took a circuitous path eventually return to the same functional state as an optimal run? While final task success flags binary outcomes, it fails to explain the topological structure of intermediate errors.

To address this need, we present TRACEFORK, an interactive system for trajectory comparison and divergence localization. TRACEFORK normalizes heterogeneous execution logs into a unified task-run-step schema, aligns paired runs via a deterministic multi-modal dynamic programming engine, and visualizes the resulting execution flow as a task-rooted *inspection tree* (Figure 1). Identical execution prefixes remain merged on a common trunk, allowing analysts to skip redundant steps. Behavioral forks immediately highlight the root-cause divergence, while conservative markers flag potential state reconvergence. Rich step-level evidence—including bounding boxes, normalized DOM summaries, textual locators, adapter confidence scores, and visual frames—is accessible on demand.

Our primary contributions are:

- An interactive **inspection-tree interface** that renders paired human-agent or agent-agent trajectories side by side with synchronized multi-modal evidence;
- A standardized **task-run-step schema** and

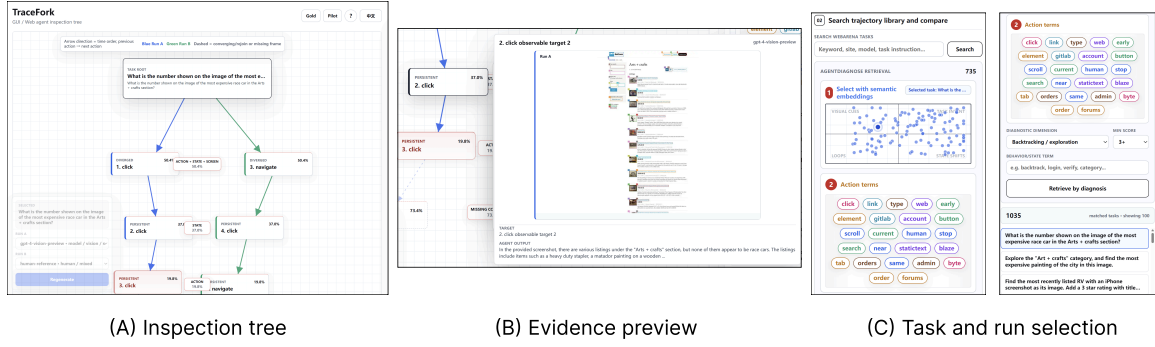


Figure 1: TraceFork inspection workflow. (A) Task-rooted inspection tree aligning two trajectories into a shared trunk, diverging branches, and potential rejoins. (B) Synchronized multi-modal evidence preview for target selectors, DOM states, model reasoning, and viewport screenshots. (C) Library navigation rail supporting diagnostic search, behavioral categorization, and trace uploads.

extensible ingestion pipeline supporting major benchmarks, Playwright sessions, and unformatted user logs;

- A fast, deterministic **alignment engine** that localizes behavioral forks and applies conservative gating for reconvergence detection;
- A comprehensive, publicly accessible **open-source artifact** containing 1,722 tasks and 1,387 aligned trajectory pairs.

Artifact availability. Interactive Web Demo: <https://yuzhang-zhong.github.io/TraceFork/>. Video Walkthrough: <https://youtu.be/rN4Jc8SFmpY>. Open-source Code and Benchmark Suite: <https://github.com/yuzhang-zhong/TraceFork>. The TRACE-FORK source code is released under the MIT License; benchmark-derived trajectories, screenshots, and other third-party artifacts remain subject to their original licenses and terms.

2 Demonstration Workflow

Figure 1 illustrates the core analyst workflow: selecting paired runs from the repository (C), inspecting the alignment tree (A), and reviewing synchronized multi-modal evidence for individual steps (B).

Tree comparison. The inspection tree follows both execution paths from the shared task instruction downward (Figure 1(A)). Identical actions remain unified on the main trunk. When agents select different targets or transition into distinct page states, the tree opens parallel branches. Extraneous intermediate actions remain on their respective

branches as visible detours. If the two runs subsequently reach functionally consistent states, the tree annotates the transition with a rejoin marker.

Step-level evidence preview. Each tree node displays its high-level action family (e.g., `click`, `type`, `navigate`, `stop`). Hovering over or selecting a node opens the evidence pane (Figure 1(B)), which renders the element locator, normalized DOM state summary, public model reasoning excerpt, adapter confidence score, and high-resolution viewport screenshot.

Curated trajectory library. The right navigation rail (Figure 1(C)) provides full-text search and behavioral filtering. Trajectories are indexed using diagnostic categories adapted from AgentDiagnose (Ou et al., 2025), including exploratory actions, state transitions, self-verification loops, and visual misgrounding. This taxonomy allows developers to isolate specific behavioral patterns without scanning the entire library.

3 Trace Ingestion and Extensibility

TRACEFORK decouples environment-specific logging from alignment and visualization through a unified intermediate schema.

Direct benchmark sources. TRACEFORK natively ingests execution dumps from WebArena and VisualWebArena, including raw execution logs, rendered HTML DOM trees, screenshot sequences, and Playwright session replays. Human demonstration recordings pass through the same normalization pipeline, preserving action timestamps, locator metadata, and visual frames.

User-provided logs and LLM normalization.

For arbitrary external agents, the system accepts custom JSON traces, standard action logs, and screenshot collections. For unformatted text logs, an optional LLM-assisted parser extracts key fields into the standardized schema. Once normalized, custom traces seamlessly integrate into the comparison interface alongside benchmark runs.

System extensibility. The frontend inspection canvas communicates with the alignment backend via a structured JSON comparison contract (Appendix B). Consequently, alternative alignment algorithms—such as Dynamic Time Warping (DTW), process mining conformance checkers, or learned embedding models—can directly drive the visual interface without modifying frontend components.

4 Comparison Engine

Given trajectory $A = (a_1, \dots, a_n)$ and trajectory $B = (b_1, \dots, b_m)$, TRACEFORK extracts multi-modal step summaries, ranks candidate correspondences, and computes an optimal deterministic alignment.

Step feature representation. For each step, the engine extracts action type, target locator string, visible text tokens, route URL, accessibility roles, and a compact visual representation comprising a 64-bit DCT perceptual hash (pHash) and normalized color-spatial histograms. Step transitions encode dynamic state changes relative to the preceding step.

Step-pair distance computation. For candidate pair (a_i, b_j) , the engine evaluates multi-field component distances d_{ij}^k . To prevent visual noise or benign selector variations from dominating alignment, each component distance is mapped through a piecewise linear neutral band:

$$e_{ij}^k = \text{clip} \left(\frac{d_{ij}^k - \alpha_k}{\beta_k - \alpha_k}, 0, 1 \right), \quad (1)$$

where α_k absorbs non-critical variance and β_k marks clear incompatibility (Appendix B).

Component scores are combined via a bounded product-union:

$$\tilde{D}_{ij} = 1 - \prod_{k \in K_{ij}} (1 - e_{ij}^k), \quad (2)$$

where K_{ij} denotes the set of features mutually observed in both steps.

To account for mission-critical discrepancies, TRACEFORK applies a hard gating penalty F_{ij} :

$$D_{ij} = \max(\tilde{D}_{ij}, F_{ij}). \quad (3)$$

The gate F_{ij} triggers on terminal stop/failure mismatches, incompatible action families, or confirmed route divergence.

Dynamic programming alignment. Global trajectory alignment is formulated as a dynamic program over alignment cost matrix C_{ij} with affine-like gap penalties:

$$M_{i,j} = \min \{ M_{i-1,j-1} + C_{ij}, \\ M_{i-1,j} + G_A(a_i, \ell_A), \\ M_{i,j-1} + G_B(b_j, \ell_B) \}, \quad (4)$$

where ℓ_A, ℓ_B denote consecutive gap lengths. Setting gap opening costs higher than gap extension costs encourages the engine to group contiguous detours into single, coherent branches rather than fragmented alternating steps. Ties are resolved deterministically by favoring higher match counts, lower cumulative divergence, and diagonal steps. The full-matrix dynamic program runs in $O(nm)$ time and uses $O(nm)$ memory when backpointers are retained for traceback. Across our dataset, alignment requires a median of 45.5 ms per pair (mean 89.2 ms, p95 320.4 ms, worst-case 1245.4 ms).

Conservative rejoin policy. In visual debugging, presenting a false reconvergence is significantly more harmful to an analyst than missing an ambiguous match. Therefore, TRACEFORK applies a deliberately conservative multi-condition filter before flagging a rejoin: candidate pairs must satisfy $D_{ij} \leq 0.22$, state distance ≤ 0.25 , local contextual similarity ≥ 0.70 , zero intervening terminal failures, and subsequent stability (either two subsequent low-drift matches or a shared terminal state). These markers are designed to serve as high-precision exploratory cues rather than absolute functional equivalence proofs.

5 Evaluation and Analysis

We evaluate the alignment quality, system robustness, computational overhead, and analyst utility across multiple benchmarks.

Case study: Cascading divergence. Consider a paired VisualWebArena task where a human

demonstrator and a model agent attempt to configure a product variant. Both traces exhibit identical step lengths and click sequences (category $\rightarrow$ product $\rightarrow$ options). In standard linear logs, all model steps appear plausible in isolation. In TRACEFORK’s tree view, the trunk splits cleanly when the human selects the target size while the agent defaults to an incorrect style option. The tree exposes the style selection as the root-cause failure, correctly framing all subsequent checkout actions as consequences of that initial mistake.

E1: Step-level annotation pilot. To assess alignment accuracy, we annotated 81 step pairs across 13 VisualWebArena tasks. Three independent annotators assigned one of six relational labels (`overlap`, `rejoined`, `diverged`, `missing_a`, `missing_b`, `unsure`). Majority vote over the three annotations yields one relation per step pair; for binary evaluation, the labels are mapped into CONVERGE ($N = 9$) and DIVERGE ($N = 72$). This exploratory pilot was kept completely distinct from heuristic configuration.

Against the aggregated labels, TRACEFORK matches 68 of 81 step pairs (accuracy = 0.840). For divergence localization, the system reaches a precision of 0.904, recall of 0.917, and F1 of 0.910. For reconvergence preservation, precision is 0.250, recall is 0.222, and F1 is 0.235, yielding a balanced accuracy of 0.570 (Figure 2).

We benchmark TRACEFORK against three standard deterministic baselines: index-only alignment, action-only Needleman–Wunsch, and action+target Needleman–Wunsch alignment (Needleman and Wunsch, 1970). While the action+target baseline achieves a slightly higher divergence F1 (0.941 vs. 0.910), it does so by aggressively treating all non-identical steps as permanent forks. As a result, its converge recall is exactly 0.000, and its balanced accuracy collapses to 0.500 (equivalent to trivial majority guessing). In contrast, TRACEFORK strikes a practical balance: maintaining high divergence localization accuracy while retaining non-zero sensitivity to trajectory rejoins.

E2: Sensitivity and field ablation. A robustness audit across 240 trajectory pairs reveals that locator and target text contribute most significantly to alignment stability (removing target text drops alignment F1 to 0.25; Appendix A). Visual frame hashes provide secondary stabilization on multi-modal tasks but are more sensitive to dynamic view-port re-renderings.

Evaluation Dimension	Summary
Corpus coverage	7 benchmark collections; 1,722 tasks; 2,933 runs; 33,337 steps; 1,387 same-task pairs; 22,205 aligned step pairs.
Pair distributions	587 human–model, 800 model–model pairs; 1,155 WebArena, 232 VisualWebArena pairs.
Annotation pilot	81 annotated step pairs; system–gold accuracy 0.840; divergence F1 0.910; converge F1 0.235; balanced accuracy 0.570.
Baseline comparison	Index-only F1 0.791; action-only F1 0.787; action+target F1 0.941 (converge recall 0.000, balanced accuracy 0.500).
Robustness checks	Node/edge bounds, schema conformance, and rendering sanitization verified on 240 audited pairs.
Computational latency	Median 45.5 ms; mean 89.2 ms; p95 320.4 ms; peak heap memory 427.9 MB.

Table 1: System performance, benchmark coverage, and validation metrics across the TraceFork library.

E3: Corpus-scale trajectory insights. Evaluating across 1,035 comparable task pairs reveals that human–model trajectory divergence is substantially higher than model–model divergence ($\Delta = 0.133$, bootstrap 95% CI [0.120, 0.145]). Interestingly, trajectory divergence between pairs with differing task outcomes was nearly identical to pairs with matching outcomes ($\Delta = -0.004$, CI [−0.030, 0.026]), confirming that trajectory divergence captures behavioral execution paths independent of whether an agent stumbled into an accidental final success.

E4: Runtime and system overhead. System runtime benchmarking over all 1,387 real pairs confirms that TRACEFORK maintains interactive responsiveness: median per-pair execution latency is 45.5 ms, with peak heap memory under 428 MB, supporting interactive use in local browser environments.

E5: Qualitative formative walkthrough. To gather early qualitative feedback on interface affordances, three experienced agent researchers participated in a formative walkthrough across 32 task scenarios. Participants compared the tree visualization against standard dual linear logs on 5-point scales measuring diagnostic confidence, perceived usefulness, and mental effort.

The linear log view received high baseline scores (confidence 4.94, usefulness 4.50, effort 1.44) compared to TRACEFORK (confidence 4.47, usefulness 4.29, effort 1.65). Qualitative feedback revealed

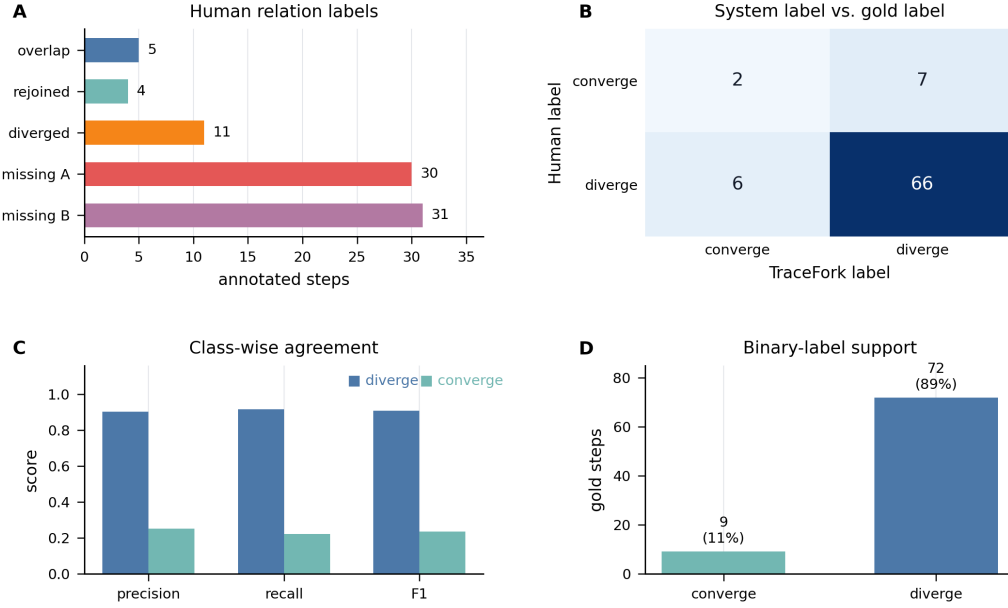


Figure 2: Human annotations and TraceFork predictions on the 81-step pilot subset ($N = 81$). The class distribution reflects the real-world dominance of divergent paths in uncured web agent traces.

that participants relied on linear logs out of established debugging habits for short, isolated actions. However, for multi-step cascading errors, participants emphasized that TRACEFORK’s tree representation provided superior structural clarity, eliminating the tedious manual back-and-forth required to isolate the original point of divergence.

6 Conclusion

TRACEFORK provides an open-source, interactive inspection workbench that transforms raw, unaligned GUI-agent traces into structured, synchronized visual trees. By automating the localization of behavioral forks and presenting contextual evidence side by side, the system helps developers diagnose root-cause failures in complex agent trajectories. The modular architecture and publicly released benchmark suite establish a foundation for trajectory debugging and comparative agent evaluation.

7 Limitations

State equivalence heuristics and hidden states. TRACEFORK’s default engine computes state similarity based on observable DOM tokens, accessibility roles, visual hashes, and locator strings. Consequently, it may miss latent, task-critical functional state transitions that lack prominent visual representations—such as toggled checkboxes, hidden authentication tokens, programmatic tab ac-

tivations, or modified background cookies. We intentionally chose this deterministic, rule-based approach to provide a lightweight, reproducible, zero-inference baseline. However, because our architecture decouples alignment computation from visualization via a standardized JSON contract, future backends can readily plug in deep DOM diffing or learned multi-modal state embeddings.

Rejoin sensitivity and evaluation scope. As demonstrated in the pilot evaluation, the current rejoin policy prioritizes precision over recall to prevent confusing visual bridges, resulting in a conservative rejoin F1 of 0.235. Furthermore, our manual annotation subset ($N = 81$) serves as a formative pilot rather than an exhaustive benchmark. A comprehensive, fully adjudicated evaluation across diverse web domains remains an important area for future community benchmark creation.

Algorithmic scalability. The current dynamic programming engine aligns trajectory pairs in $O(nm)$ time, which executes comfortably within 100 ms for standard benchmark horizons (≤ 50 steps). Scaling to long-horizon workflows or simultaneous multi-run consensus graphs will require banded DP approximations, landmark anchor indexing, or consensus tree abstractions.

Ethics Statement

Data privacy and redaction. TRACEFORK operates on public benchmark datasets or sanitized user uploads. Because web navigation traces can inadvertently record sensitive data (such as authentication tokens, passwords, cookies, or personal browsing history), the ingestion pipeline includes automated client-side sanitization to redact credentials, local file paths, and private URLs. The system stores only bounded thumbnail representations and anonymized locator strings.

Interpretation of alignment cues. A high divergence score indicates that two execution paths followed different observational sequences; it does not inherently imply that one path is erroneous. Developers must review the synchronized multimodal evidence before drawing causal conclusions regarding model capabilities or task success.

References

- Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2Web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*.
- Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. 2024. WorkArena: How capable are web agents at solving common knowledge work tasks? Preprint, arXiv:2403.07718.
- Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. 2024. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*.
- Thibault Le Sellier de Chezelles, Maxime Gasse, Alexandre Drouin, Massimo Caccia, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omid Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Quentin Cappart, Graham Neubig, Ruslan Salakhutdinov, Nicolas Chapados, and Alexandre Lacoste. 2024. The BrowserGym ecosystem for web agent research. Preprint, arXiv:2412.05467.
- Saul B. Needleman and Christian D. Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453.
- Tianyue Ou, Wanyao Guo, Apurva Gandhi, Graham Neubig, and Xiang Yue. 2025. AgentDiagnose: An open toolkit for diagnosing LLM agent trajectories. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 207–215.
- Siva Reddy, Xing Lu, and Zdeněk Kasner. 2024. Weblinx: Real-world website navigation with multi-turn dialogue.
- Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. 2024. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems*.
- 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. 2024. WebArena: A realistic web environment for building autonomous agents. In *Proceedings of the Twelfth International Conference on Learning Representations*.

A Comparison Configuration

This appendix records the deterministic parameter configuration and robustness sensitivity results.

Table A1: Results from four robustness checks.

Test	Result	What we found
Index alignment / action LCS	Action LCS places the first fork within one step in most cases. Full alignment F1 is ≈ 0.13 .	Action sequences provide coarse alignment; target locators and page states are essential for step-level correspondence.
Target text removed	Removing target text causes the largest degradation: F1 0.25 and $\Delta D = 0.147$.	Target names and selector text provide the strongest anchor signal for step matching.
Boxes or frames removed	Alignment F1 remains 0.84 without bounding boxes and 0.71 without visual frames.	Textual DOM features carry the majority of structural signal; visual hashes assist primarily on multimodal tasks.
Step dropout / missing fields	First fork localization remains stable under 20% step dropout; F1 drops to 0.32 under multi-field omission.	Affine gap modeling absorbs occasional missing steps, but multi-modal missingness degrades alignment quality.

Note. Robustness audit evaluated on 240 benchmark pairs. F1 values here measure step-correspondence agreement with the full-feature alignment and are distinct from the human-gold divergence F1 reported in Section 5.

Gating thresholds and tie-breaking. Hard gate penalties F_{ij} are configured as: 0.80 for terminal status mismatches, 0.70 for conflicting action categories, 0.65 for high-confidence target mismatches, and 0.60 for route/title transition conflicts. DP tie-breaking deterministically prioritizes: maximum matched pairs, minimum cumulative divergence, fewest triggered hard gates, minimum index displacement, and diagonal transitions.

B Comparison Artifact and Extension Points

Table B1: Modality signals and neutral distance bands.

Signal	Distance Formulation	Neutral Band (α_k, β_k)
Action	Action taxonomy distance (exact match, within-category, cross-category conflict).	(0.00, 0.45)
Target	Normalized ARIA attributes, selector endings, and label text via token-weighted Jaccard distance.	(0.10, 0.60)
Region	Bounding-box intersection over union ($1 - \text{IoU}$) and centroid displacement.	(0.10, 0.50)
State	Normalized URL route tokens, headings, and accessibility trees via weighted n-gram Jaccard.	(0.12, 0.55)
Transition	Inter-step token differentials and action group dynamics via Jaccard distance.	(0.10, 0.50)
Visual	Viewport 64-bit DCT pHash distance combined with color-spatial layout histograms.	(0.08, 0.35)

JSON Comparison contract. The system schema specifies trajectory metadata, aligned step pairs, one-sided gap segments, multi-component

distance breakdowns, gate activations, and optional reconvergence pointers. This contract isolates the rendering client from the alignment engine, facilitating third-party aligner integration.