VisualTreeSearch: Understanding Web Agent Test-time Scaling

Danqing Zhang, Yaoyao Qian, Shiying He, Yuanli Wang, Jingyi Ni, Junyu Cao
VisualTreeSearch-Demo, Demonstration Video

PathOnAI.org, Northeastern University, Boston University, The University of Texas at Austin

Abstract. We present VisualTreeSearch, a fully-deployed system for visualizing and understanding web agent test-time scaling. While test-time search algorithms substantially improve web agent success rates, they remain confined to research contexts with limited practical deployment. Our system bridges this gap with three key contributions: (1) a production-ready solution with cloud-based architecture, (2) an efficient API-based state reset mechanism that reduces state reset time from 50 to 2 seconds, and (3) an interactive web UI that transparently demonstrates the agent's decision-making process. VisualTreeSearch provides an intuitive framework for both researchers and users to understand tree search execution in web agents.

Keywords: Web Agents · Tree Search · Vision-Language Models · Test-time Scaling · Interactive Visualization.

1 Introduction

Recent years have witnessed significant advancements in autonomous web agents powered by LLMs and VLMs for browser automation, enhancing human-computer interaction by executing complex tasks from natural language instructions [10].

VLM-based Web/GUI Agent Architectures. Current VLM-based frameworks typically implement a two-phase sequential approach: action generation followed by action grounding. The action generation phase employs either VLM-based policies with carefully engineered prompts [8] or specialized purpose-built models [1,6]. For action grounding, web agents utilize website structural features [8], whereas GUI agents primarily rely on visual grounding techniques [3,6].

VLM-based Agent Test-time Scaling. Since late 2024, researchers have explored test-time scaling methodologies using search algorithms (BFS, DFS, MCTS) [2,7] and reinforcement learning [4,5] to improve web agent performance. Despite promising results, test-time scaling approaches remain confined to research contexts with limited practical deployment.

VisualTreeSearch addresses several critical gaps in existing research by providing an end-to-end solution for understanding web agent test-time scaling:

1. *Fully-deployed web agent tree search system:* Our production-ready solution includes AWS ECS services for backend and browser operations, a Vercel-hosted frontend, and CI/CD pipeline integration. The source code is fully open-sourced.

2. *Fast state reset mechanism:* We solve the critical problem of persistent website states during backtracking through an API-based account reset method that reduces reset time from 50 to 2 seconds, enabling accurate trajectory evaluation during tree search.
3. *Interactive Visualization Interface:* Our web UI demonstrates agent decision-making through D3.js tree visualizations, live browser interfaces, and execution logs.

This allows non-technical users to understand test-time scaling, while enabling researchers to deploy their web agents for demonstration purposes.

2 Demonstration

2.1 High level overview

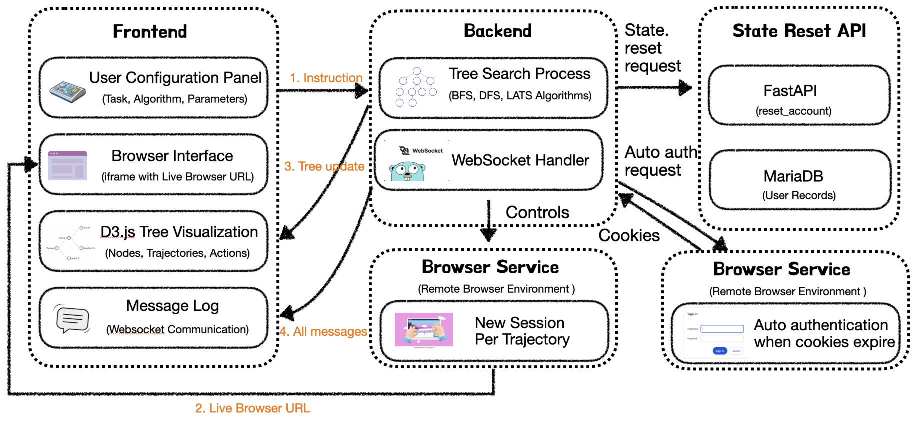


Fig. 1: System Design: High Level Overview.

Figure 1 provides a high-level overview of our implemented system. Our web agent tree search visualization system consists of four main components working together:

– **State Reset API**: A specialized service provides an efficient state reset mechanism. It enables web agents to restore a clean initial state before starting each new trajectory. This prevents evaluation inconsistencies caused by persistent website state changes.
– **Backend**: A backend service that implements various tree search algorithms and manages real-time WebSocket communication with the frontend to transmit agent execution information.
– **Browser Service**: A remote browser service that provides isolated browser sessions where web agents can execute actions, while also managing automatic authentication using Playwright.
– **Frontend**: Provides the user interface for configuring search tasks, visualizing tree search trajectories, and observing agent behavior through embedded browser views and execution logs.

2.2 API-based state reset

When web agents interact with UIs, they modify states that persist in the website's database, causing evaluation inconsistencies across trajectories. This state persistence creates scoring inaccuracies, as one trajectory may incorrectly include website state changes from previous trajectories. Our solution implements an API-based state reset mechanism with a FastAPI server hosted on AWS EC2 that manages the website database to control website state (MariaDB for our demo), reducing reset time from **50** seconds with previous docker container restarts to just **2** seconds.

2.3 Backend

In our backend, we implement several tree search algorithms like BFS, DFS and MCTS variants like LATS[9] as examples. Unlike previous Vercel-based web agent demo [8], our system implements AWS ECS container-based services to overcome Vercel's serverless execution limitations. This architecture supports persistent WebSocket communication and accommodates extended processing times, both of which are essential for comprehensive tree search operations.

2.4 Browser Service

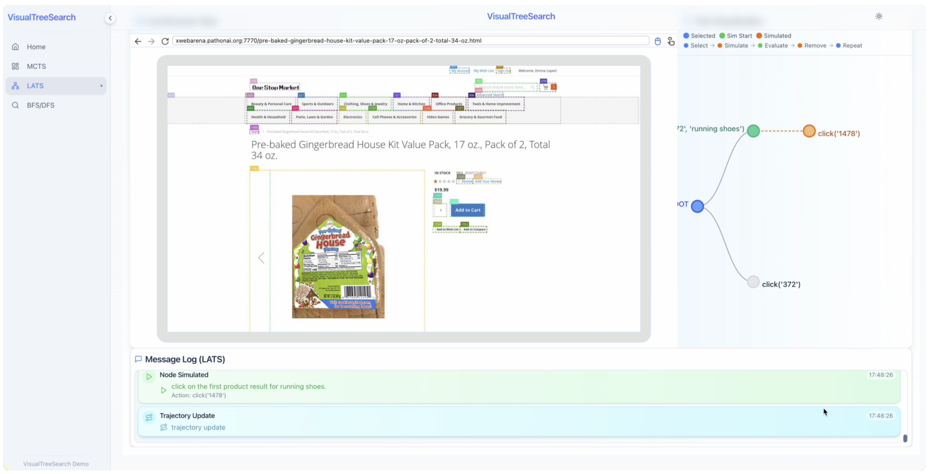
For browser integration, our primary solution employs BrowserBase for remote browser sessions, utilizing session identifiers to maintain connection continuity and render live browser interactions in the frontend interface. However, we encountered CAPTCHA challenges with our BrowserBase hobby account during automated authentication. To address this limitation, we deployed a custom Docker-based browser service on Amazon ECS running Chromium.

2.5 Frontend

The VisualTreeSearch frontend visualization system enhances web agent research by providing an interpretable monitoring environment. As shown in Figure 2, the system comprises three main components for observing agent decision-making. The main visualization area features: (1) Browser Interface: Real-time web environment view through BrowserBase integration. (2) Tree Visualization: Hierarchical D3.js visualization highlighting the active trajectory, with interactive nodes providing action descriptions and execution outcomes. (3) Execution Log (Message Log): Chronological communication log documenting action generation requests, grounding processes, execution commands, and status updates, providing comprehensive insight into the agent's operational sequence.

3 Conclusion

This paper introduces VisualTreeSearch, a system that deploys web agent test-time scaling techniques in a production environment with a transparent, interactive visualization interface. To the best of our knowledge, VisualTreeSearch is


VisualTreeSearch
Home
MCTS
LATS
BFS/DFS
VisualTreeSearch Demo
VisualTreeSearch
xwebarena.pathonai.org:7770/pre-baked-gingerbread-house-kit-value-pack-17-oz-pack-of-2-total-34-oz.html
One Stop Market
Pre-baked Gingerbread House Kit Value Pack, 17 oz., Pack of 2, Total 34 oz.
IN STOCK
$19.99
Add to Cart
● Selected ● Sim Start ● Simulated
● Select → ● Simulate → ● Evaluate → ● Remove → ● Repeat
'72', 'running shoes')
click('1478')
OT
click('372')

Message Log (LATS)

Node Simulated 17:48:26
click on the first product result for running shoes.
Action: click('1478')

Trajectory Update 17:48:26
trajectory update

Fig. 2: Screenshot of the frontend UI showing browser interface, tree visualization and execution log.

the first such framework for web agents. By enabling efficient tree search execution in real-world web environments, VisualTreeSearch helps democratize these advanced techniques for broader use. The open-source implementation serves as both a demonstration tool and a foundation for future research on agent decision-making optimization. As web agents advance, robust visualization frameworks will be essential for developing more reliable autonomous systems.

References

1. Claude computer use model, 2024.
2. Jing Yu Koh et al. Tree search for language model agents. *arXiv preprint arXiv:2407.01476*, 2024.
3. Yadong Lu et al. Omniparser for pure vision based gui agent. *arXiv preprint arXiv:2408.00203*, 2024.
4. Zhengxi Lu et al. Ui-r1: Enhancing action prediction of gui agents by reinforcement learning. *arXiv preprint arXiv:2503.21620*, 2025.
5. Pranav Putta et al. Agent q: Advanced reasoning and learning for autonomous ai agents. *arXiv preprint arXiv:2408.07199*, 2024.
6. Yujia Qin et al. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025.
7. Xiao Yu et al. Exact: Teaching ai agents to explore with reflective-mcts and exploratory learning. *arXiv preprint arXiv:2410.02052*, 2024.
8. Danqing Zhang et al. Litewebagent: The open-source suite for vlm-based web-agent applications. *arXiv preprint arXiv:2503.02950*, 2025.
9. Andy Zhou et al. Language agent tree search unifies reasoning, acting, and planning in language models. In *Forty-first International Conference on Machine Learning*.
10. Shuyan Zhou et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.