GitHub Repo Release Info Pulling - Observations

Steven Pan

2026-02-01

Iterative Refinement of Target Selection Logic

Initial Approach & Visual Challenges Initially, I implemented a bordered selection box on screenshots and prompted a multimodal model to "move the selection box to include the target." The model was provided with the following function-calling tools:

- **Translate Box:** Moves the box in four directions by fixed increments.
- **Zoom Box:** Scales the box up or down by fixed increments.
- **Interactions:** Executes keyboard inputs, clicks, and double-clicks.

This approach yielded no progress. I suspected the model struggled to perceive the thin selection border, but even after switching to a high-contrast red shaded rectangle, performance remained poor.

Transition to Agentic Loops To improve reliability, I decomposed the problem into smaller tasks by introducing an agentic loop. The system now constantly monitors the current state to determine if the target is within the box. If the target is not captured, the logic determines the next necessary action (zooming, moving, or inputting text).

While this improved performance—correctly moving toward the target in roughly 50% of cases—the model frequently suffered from "looping," where it moved the box back to previously visited coordinates. Resetting the message history and refining the prompt offered negligible improvements.

Final Strategy: Divide and Conquer Observing that the model was significantly more accurate at **verifying** the target's presence than **manipulating** the tools, I shifted to a "divide and conquer" strategy. I restricted the model's role solely to locating the target, while the Python backend handles the search logic.

This narrowed focus allows the system to locate targets using a **logarithmic search pattern**. To mitigate hallucinations, I implemented a "zoom-out" recovery step, allowing the system to backtrack if a false positive is detected. This final iteration has proven to be the most robust.

Future Improvements

- **Dynamic Bounding Box Refinement:** Currently, target localization relies on a fixed box size. This could be improved by allowing the model to evaluate the "fit" of the selection box. By prompting the model to determine if the box captures the target precisely, the system could dynamically adjust the box's dimensions and aspect ratio to match the target’s geometry.
- **Asynchronous Execution & Parallelization:** To minimize latency, certain model interactions could be parallelized. Implementing asynchronous requests (sending multiple prompts to the model simultaneously) would reduce the waiting time between state checks and action executions, leading to a more responsive agent.

Code Quality, Testing and Errors

- Did not pay much attention to code quality due to the limited time
- The major source of error would be the model’s response
    - I’ve added guards to allow for false positives
- Not tested for other repos

Ideas on Bonus

- The current method theoretically works for any repo on GitHub (not tested)
- Ask the model to extract the name of the repo from natural language prompts
- The last screenshot contains all information about the releases, including release notes, which can be easily extracted into JSON by the model