

Reasoning in Multimodal GUI Agents: An Exploration-Driven Perspective

Dr. Chi Zhang

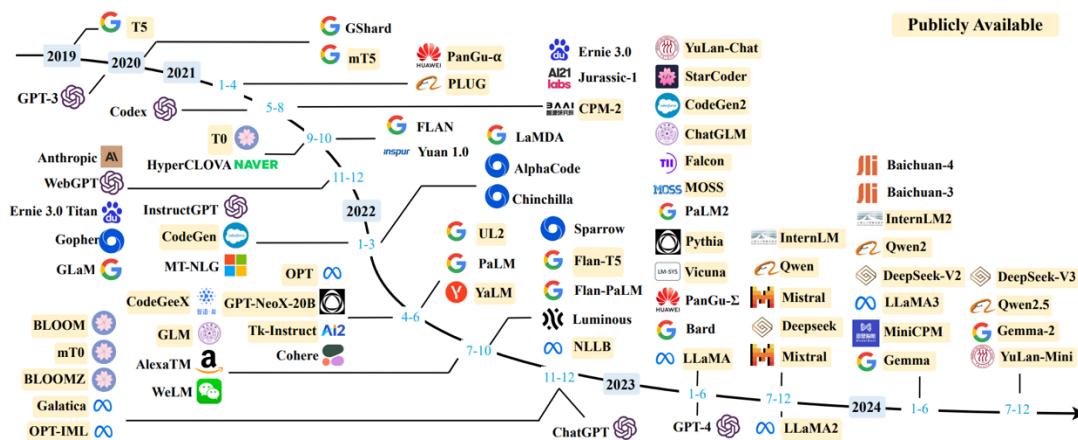
AGI Lab, Westlake University



► Background

Large Language Model

Large Language Models (LLMs) are artificial intelligence systems trained on massive amounts of data, possessing powerful capabilities in **natural language understanding and generation**. They can provide support in scenarios such as writing, coding, translation, reasoning, and scientific research.



A Survey of Large Language Models, Arxiv, 2023

How should I understand large language models?

Thought for 10 seconds ↴

🕒 Hmm, the user is asking for a conceptual understanding of large language models, which is a broad but fundamental question. They likely want more than a textbook definition—something intuitive yet comprehensive enough to grasp how LLMs work and why they matter.

I should structure this as a layered explanation, starting with simple analogies and gradually deepening the technical details. The autocomplete analogy seems like a strong entry point—it's familiar and scales well to illustrate the core idea. Then I can bridge to how this scales into a "statistical dream machine" to emphasize the emergent capabilities.

The architecture breakdown (input/output, transformer, training) is necessary but should avoid jargon overload. Focusing on the transformer's role in handling context and parallelism will help differentiate LLMs from earlier models. Finally, grounding the discussion in strengths/weaknesses and societal impact will make it practical. The goal is to balance clarity with depth, leaving the user with a coherent mental model.

Of course. Understanding Large Language Models (LLMs) can be broken down into a few key layers: a simple analogy, the core technical idea, how they are built, and their strengths and



ChatGPT



Claude



Gemini



Qwen



deepseek

► LLM as Agent

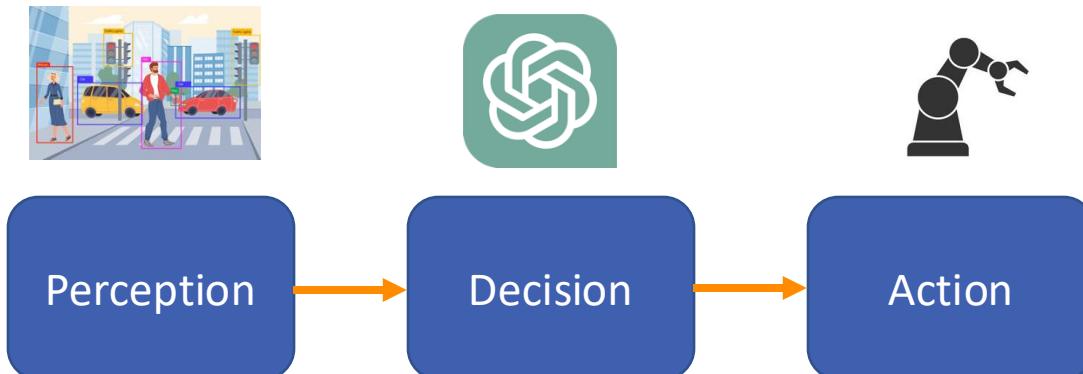
Prompt:

Now you are at the kitchen, this is what you can see from there. Can you see the fridge from here? What is your next action?



LLM:

From this position, I can see the fridge on the right side of the kitchen. My next action would be to move forward and turn slightly to the right to approach the fridge.



AI Agents are autonomous systems, powered by LLMs, designed to perceive their environment and take actions to achieve goals.

► LLM as Agent

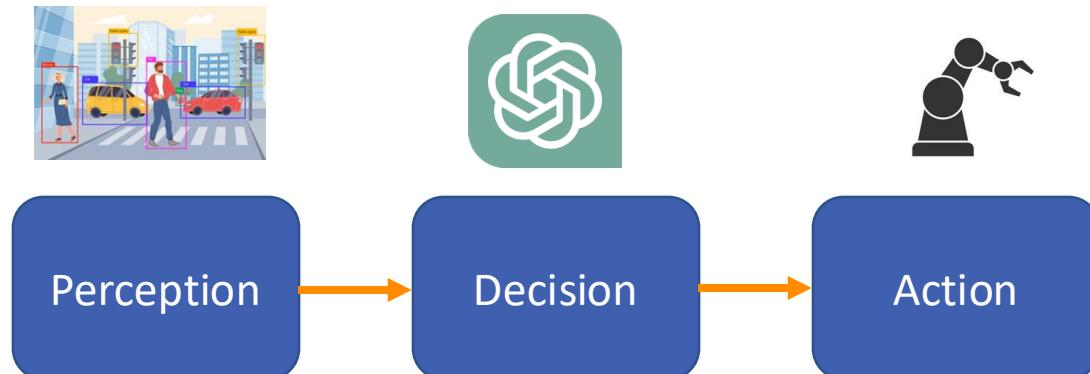
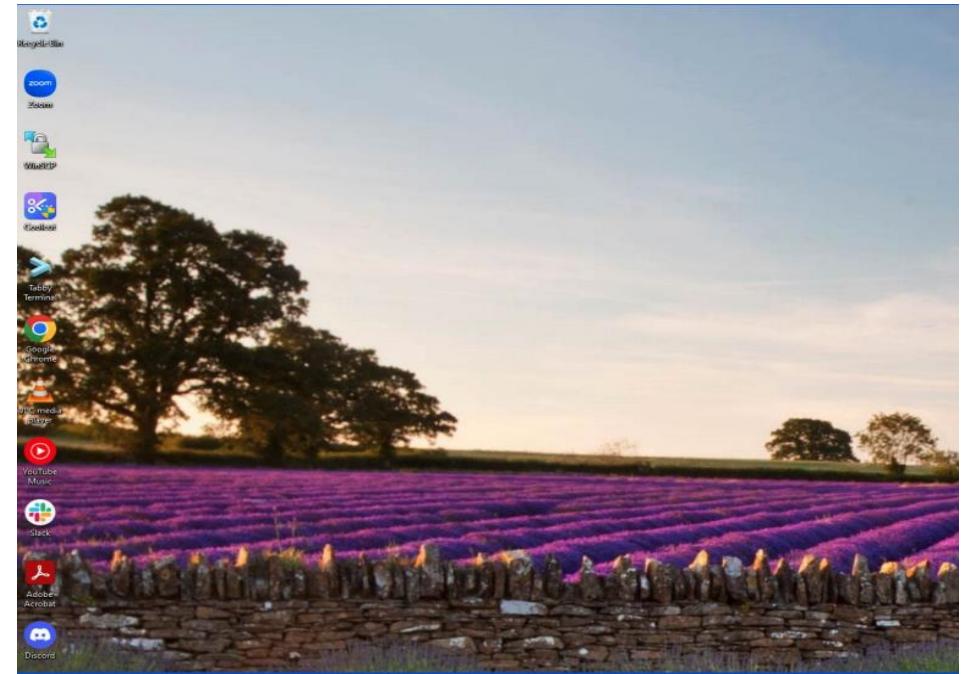
Prompt:

Imagine that you are a robot operating a computer... You are asked to find a detailed recipe of Mapo Tofu.

Below is what you see on the computer screen, predict your next move...

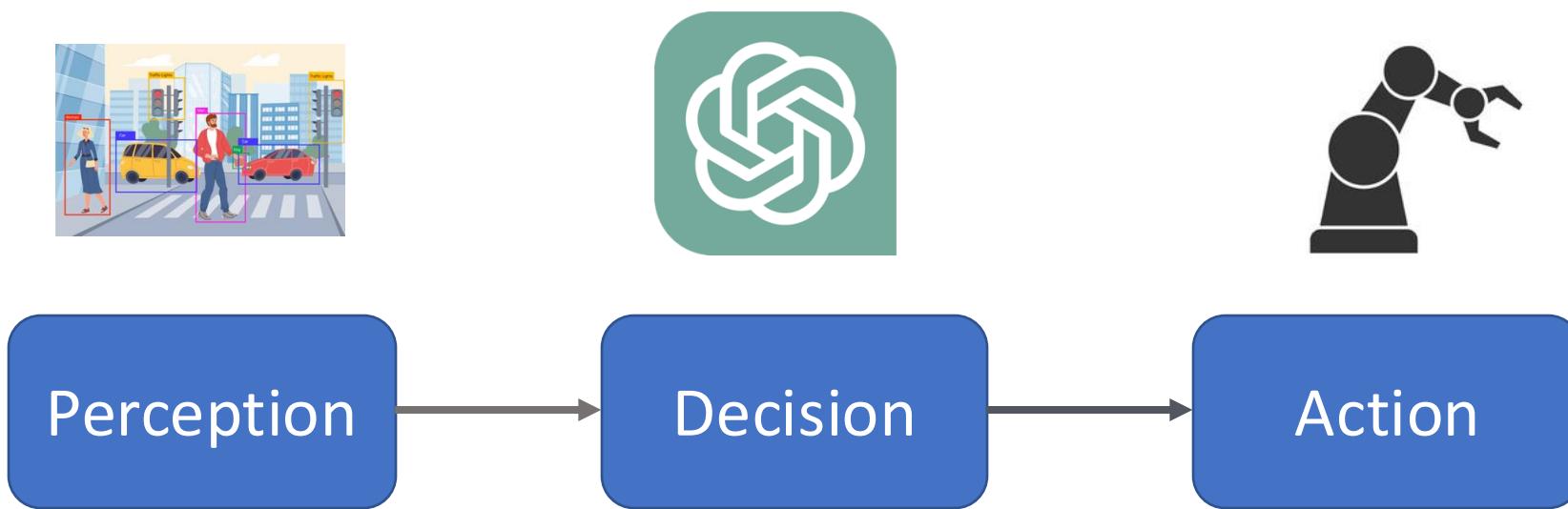
LLM:

Next move: I would **move the mouse to the Google Chrome icon** located on the left side of the screen, approximately 1/3 of the way down from the top. I would then **click on the icon** to open the web browser.



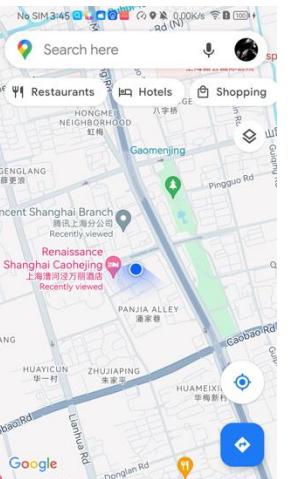
AI Agents are autonomous systems, powered by LLMs, designed to **perceive their environment** and **take actions** to achieve goals.

► A Simple Agent Prototype



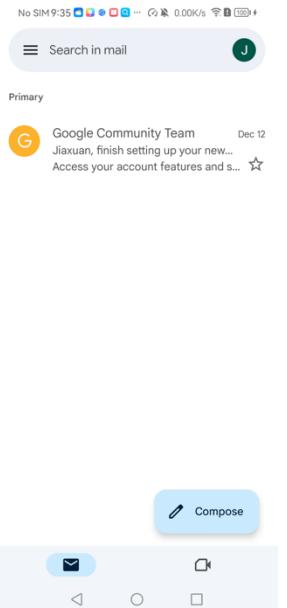
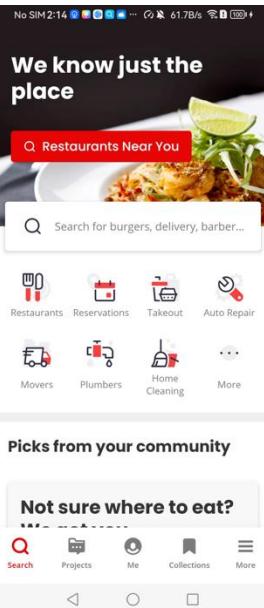
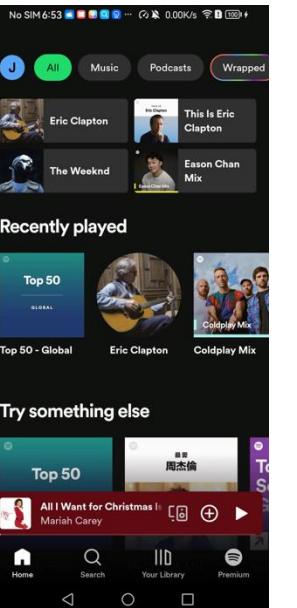
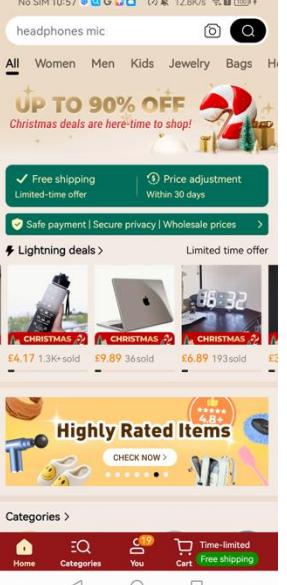
► Significance of UI





Latest in Xuhui District

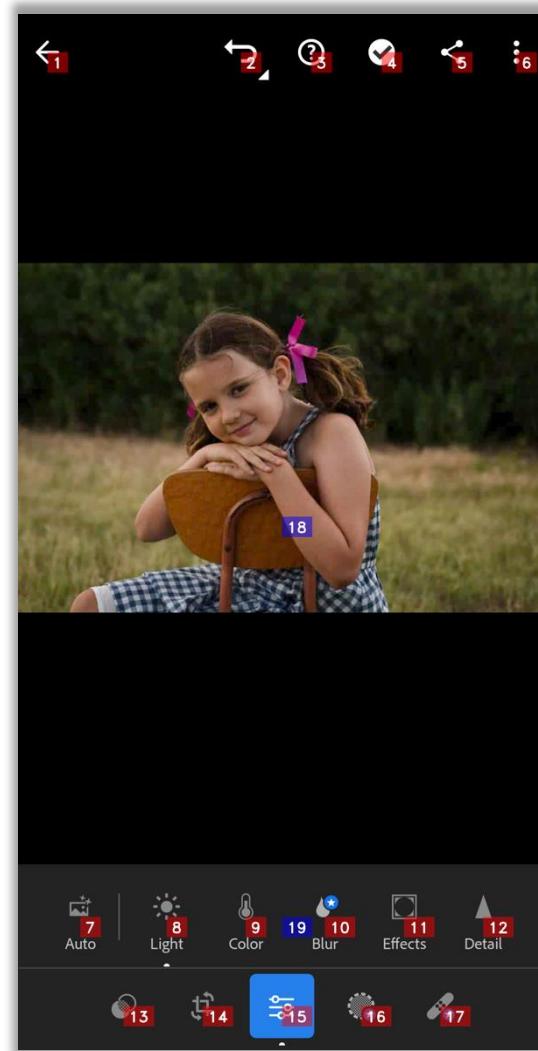
Explore Go Saved Contribute Updates



AppAgent: Multimodal Agents as Smartphone Users, CHI 2025, Zhang et al.

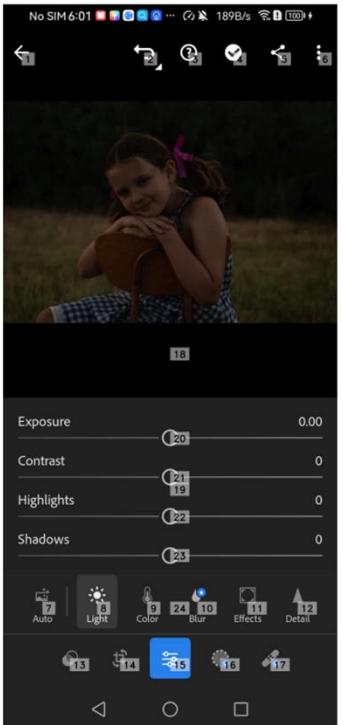
Action Space

- **Tap(element : int)**
- **Long_press(element : int)**
- **Swipe (element : int, direction : str, dist : str)**
- **Text(text : str)**
- **Back()**
- **Exit()**



► A Demo for GUI Agent

Lightroom: improve the quality of this image until it looks good to you



Observation

The screenshot shows a photo editing application interface with various adjustment settings for **exposure, contrast, highlights, and shadows**. These adjustments have **numeric tags 20, 21, 22, and 23** respectively. The settings are currently at their default values, and the image appears **underexposed**, needing improvement in lighting.

Thought

To improve the image quality and correct the **underexposure**, I need to **increase the exposure**. The **slider for the exposure setting** is labeled with **numeric tag 20**, which I can interact with to adjust the exposure level. I will **swipe right on this slider to increase the exposure**.

Action

```
swipe(20, "right", "medium")
```



Deployment Phase

Doc Source: Human Demonstration

APP: Adobe Lightroom

Task: Improve this image until it looks good to you

► Reasoning for GUI agent

LLM Reasoning

What is the output when concatenating the last letter of each word in “artificial intelligence”?

Reasoning / thinking:

The last letter of “artificial” is “l”.
The last letter of “intelligence” is “e”. Concatenating “l” and “e” leads to “le”. So the answer is “le”.

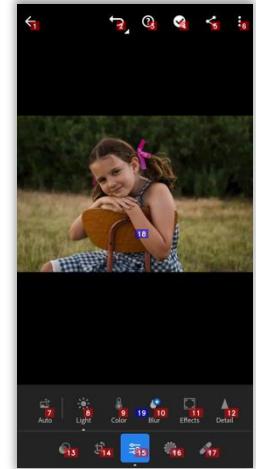
Answer:

“le”



GUI Agent Reasoning

I want to adjust this photo to make the lighting more reasonable.



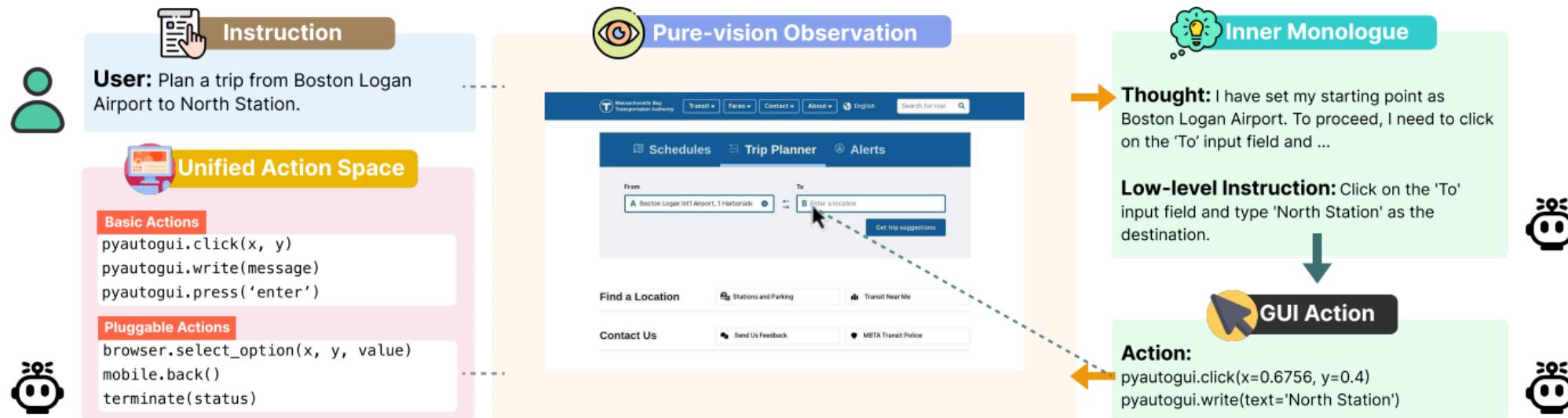
Reasoning / thinking:

I can see this is the interface of a mobile photo editing app. I'm going to try editing this photo...

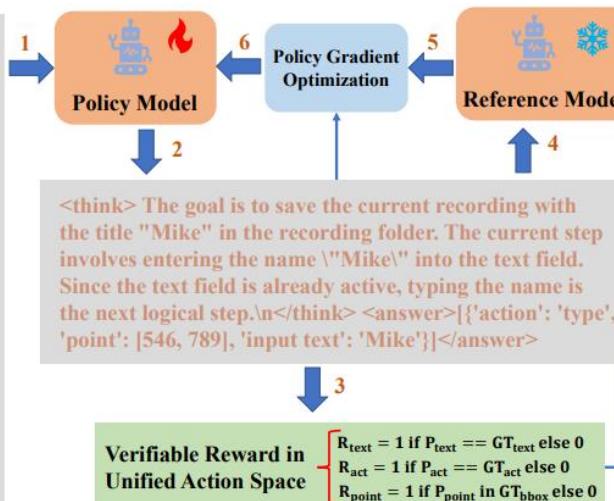
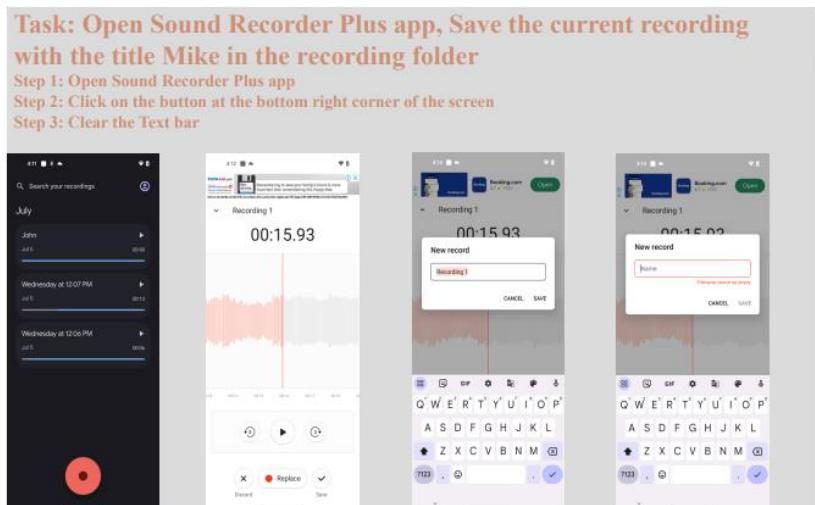
Answer:

Tap <15>

► Some Reasoning GUI Agent Works



Aguvis: Unified Pure Vision Agents for Autonomous GUI Interaction



The core goal is to enable agents to deconstruct, plan, and execute multi-step tasks. In the GUI domain, this transcends understanding a single intent and appears in various forms:

- **Generative Planning**
- **Policy Optimization**

► Reasoning for GUI agent

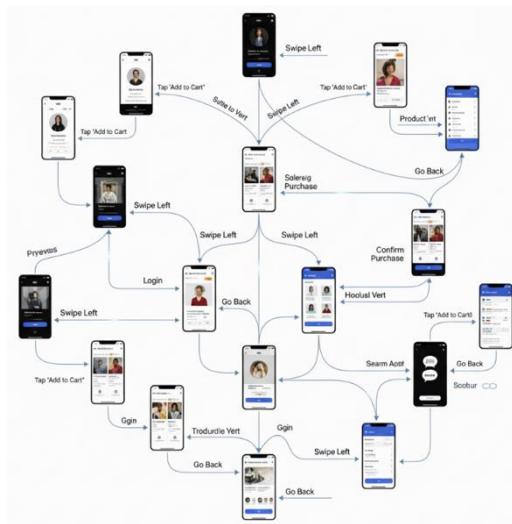
Agents need to comprehend both:

- **Application Logic** (e.g., a flight booking process)
- **UI Functionality** (e.g., where buttons are, what text fields do).

Static Reasoning:

- A “think first, act later” static planning model may not suffice.

Solution: Reasoning during Action



► Action Reasoning: Learning by Exploration

- **Mimics Human Exploration**
Learns app logic through “trial-and-error”

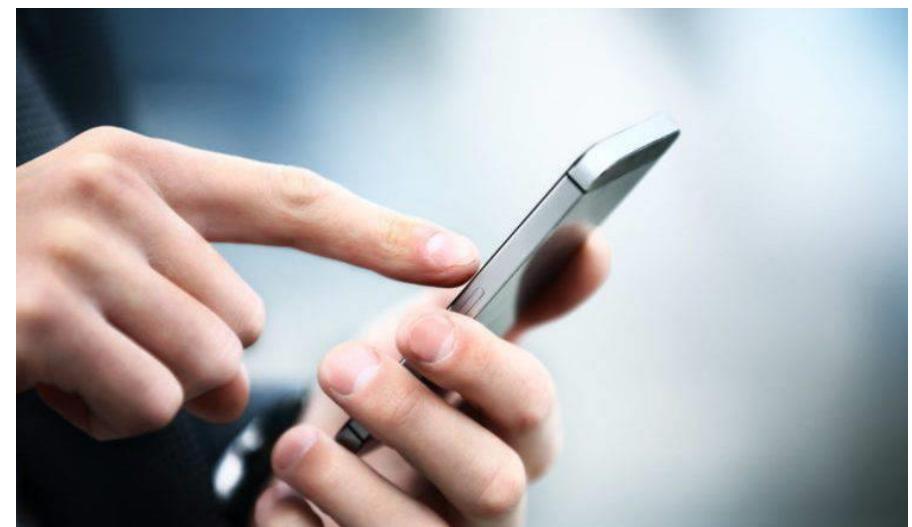
- **A Dynamic Process, Not a Static Plan**
Observe → Reason → Act → Reason



► Human-like system design

How do we human learn?

- Quick Learning process
 - Self-exploration
 - Watching demos
- Few-shot learning ability
- Good generalization



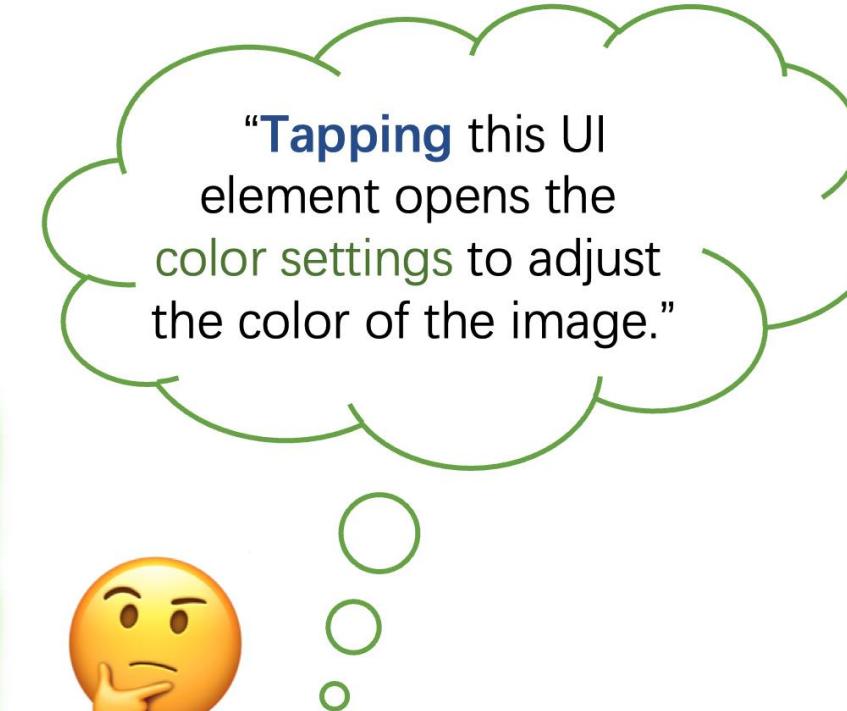
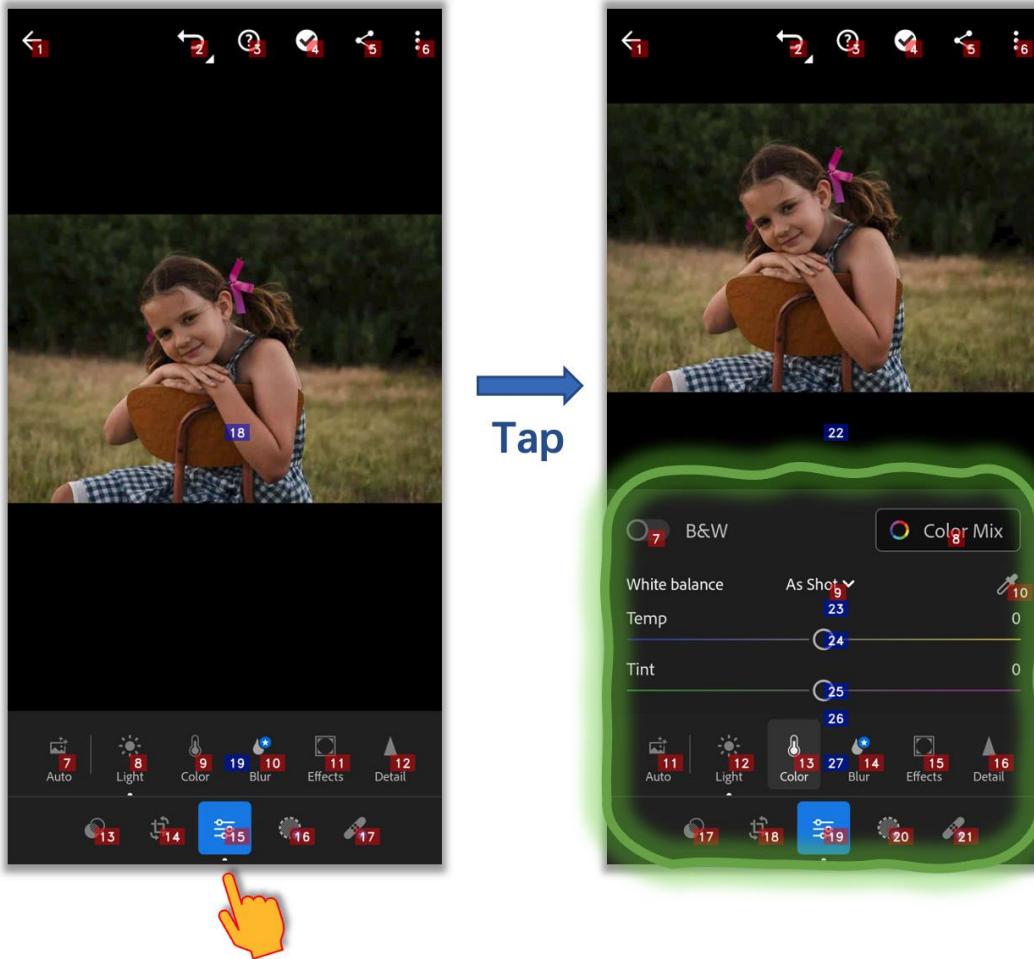
► Human-like system design

Overall Idea: Exploration>Deployment

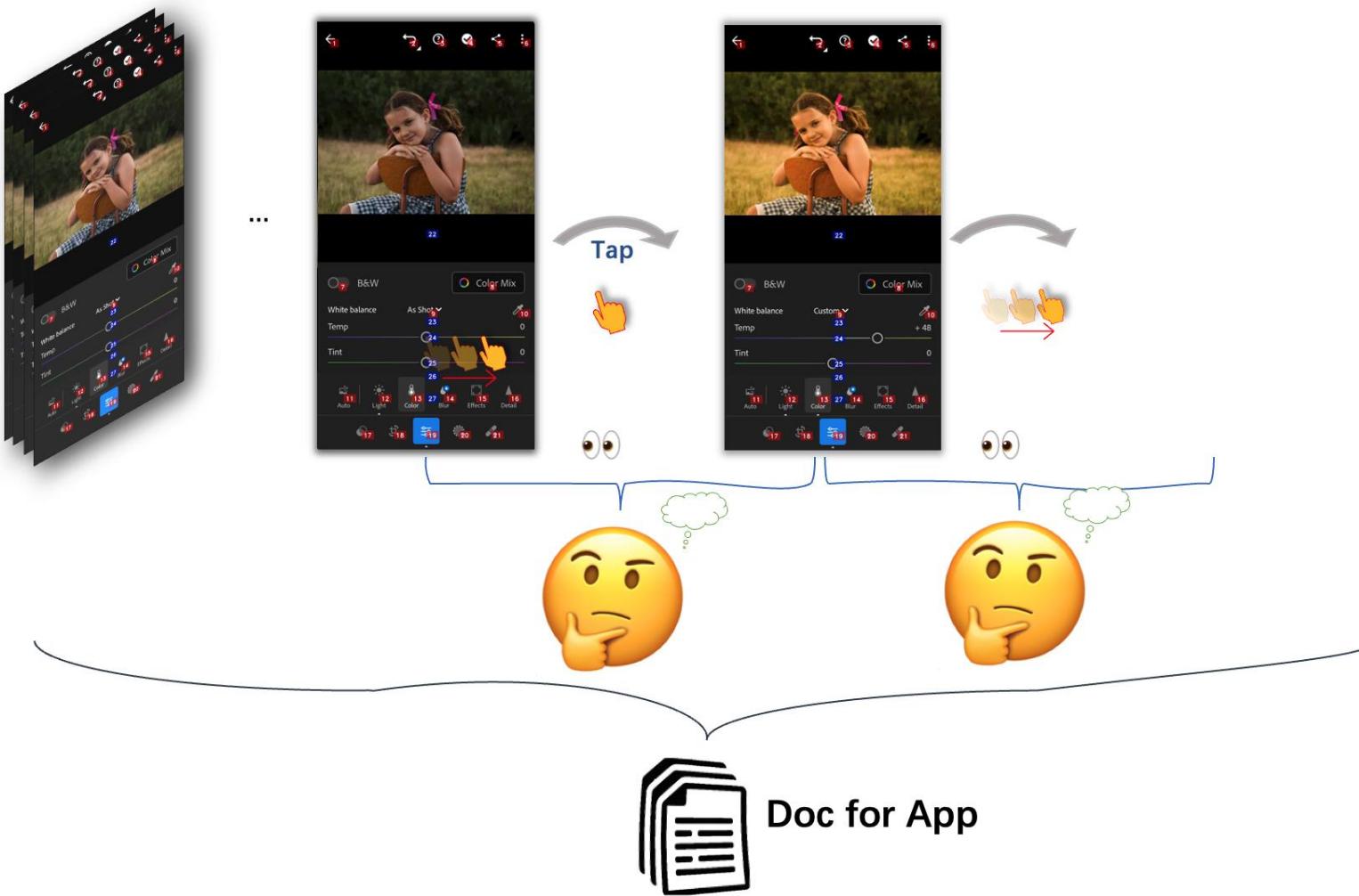
Explore how to use an App (self-exploration or watching demos)

- Generate a **document** after exploration, used as an external database
- Refer to the Doc for better decision making

► Exploration Phase



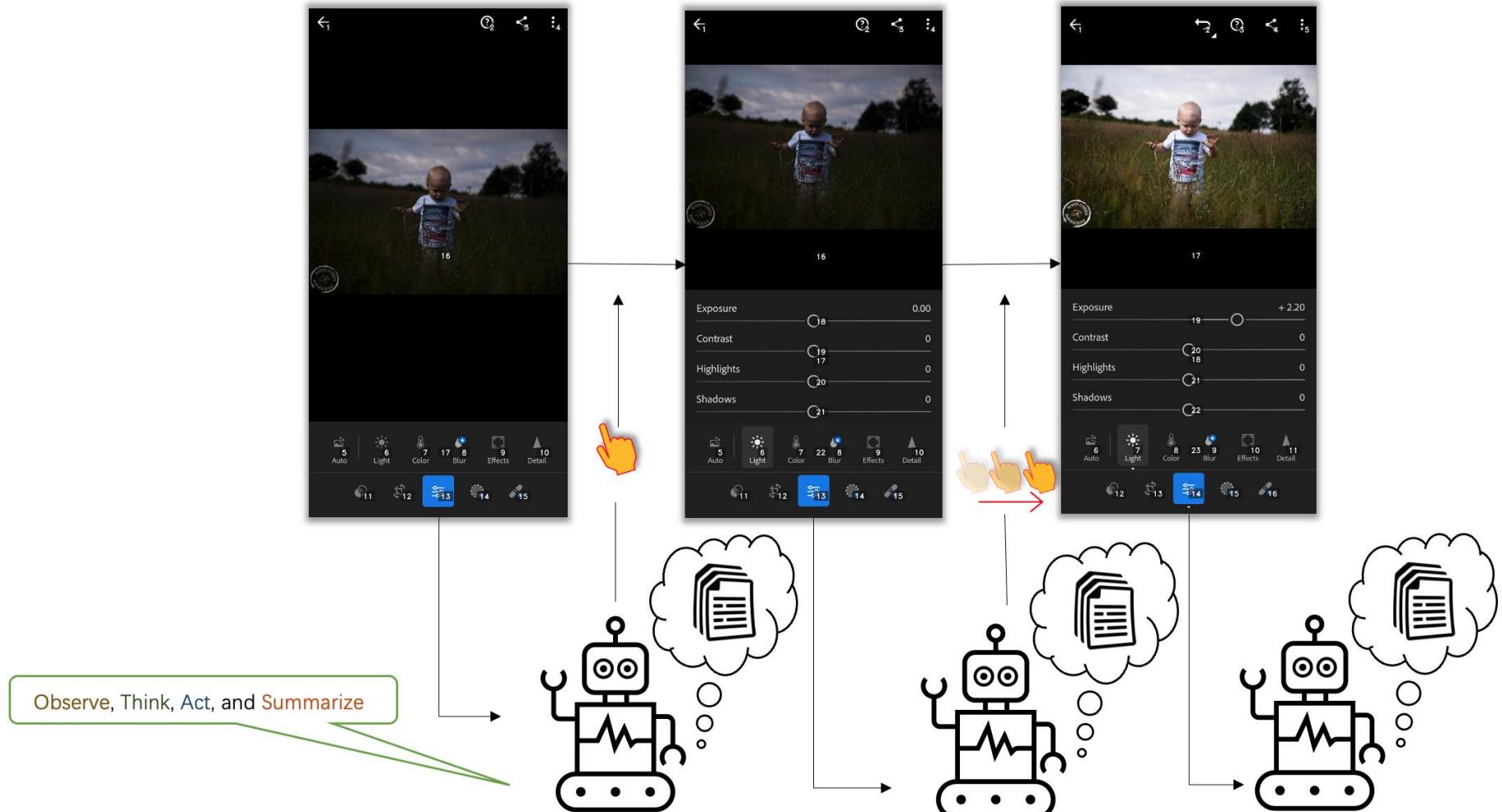
► Exploration Phase



► Deployment Phase

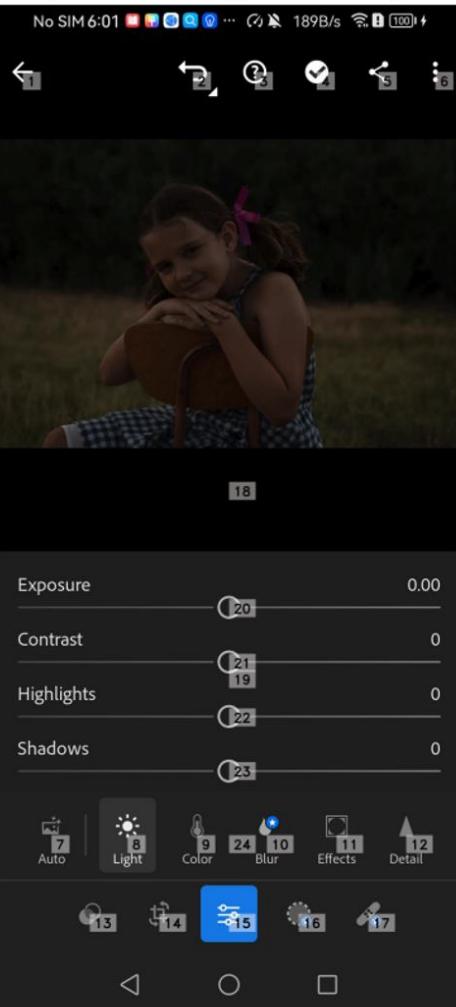
Task:

“beautify this photo...”



► Deployment Phase

Lightroom: improve the quality of this image until it looks good to you



Observation

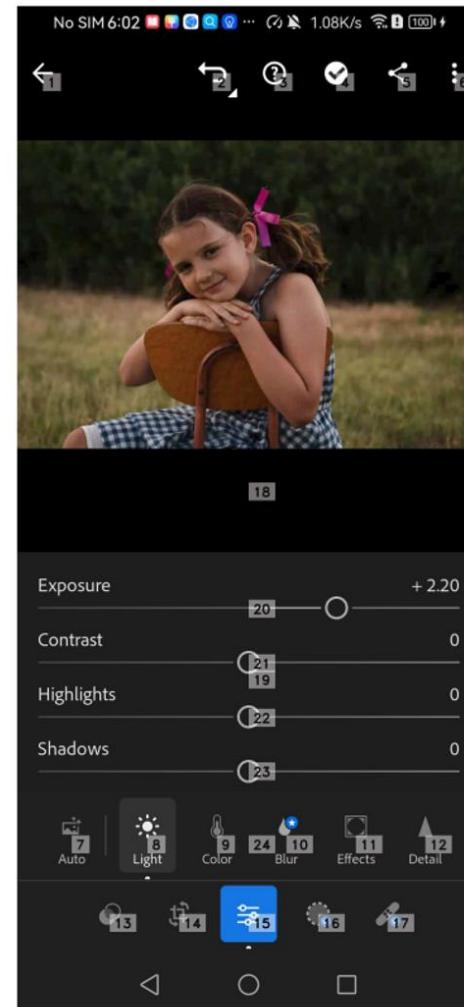
The screenshot shows a photo editing application interface with various adjustment settings for **exposure, contrast, highlights, and shadows**. These adjustments have **numeric tags 20, 21, 22, and 23** respectively. The settings are currently at their default values, and the image appears **underexposed**, needing improvement in lighting.

Thought

To improve the image quality and **correct the underexposure**, I need to **increase the exposure**. The **slider for the exposure setting** is labeled with **numeric tag 20**, which I can interact with to adjust the exposure level. I will **swipe right on this slider to increase the exposure**.

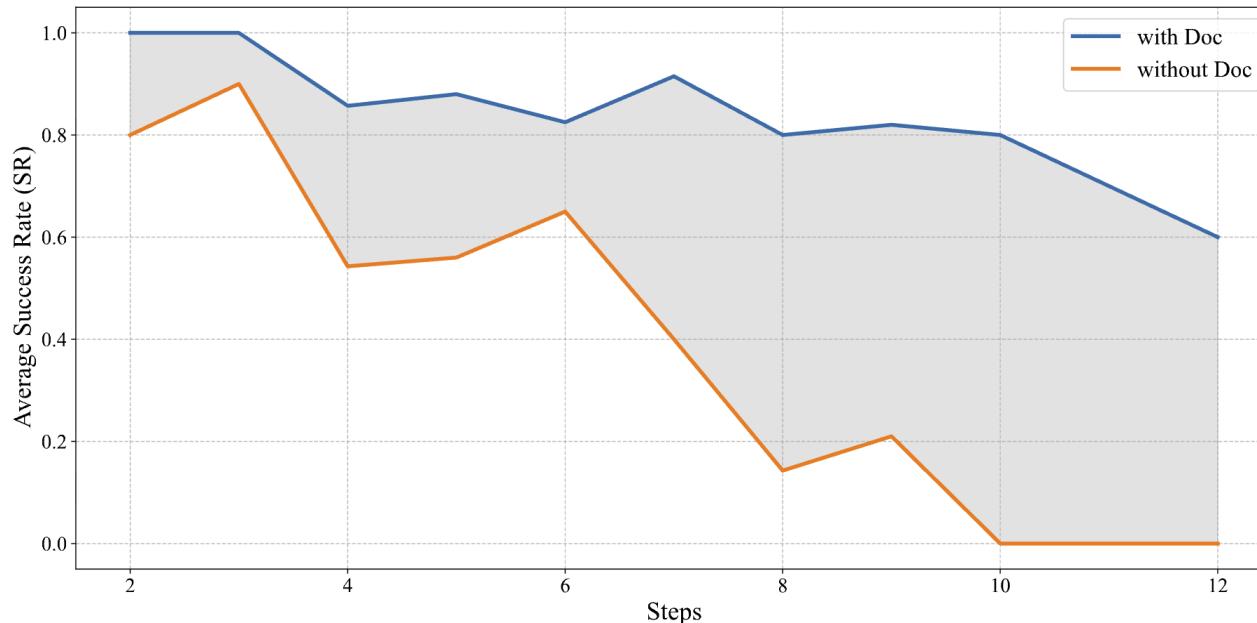
Action

`swipe(20, "right", "medium")`



Experiment

Method	Document	Action Space	SR ↑	Reward ↑	Avg. Steps
GPT4 (Baseline)	None	Raw	2.2%	0.6	4.0
	None	Ours	48.9%	3.5	6.9
AppAgent	Auto. Exploration	Ours	73.3%	5.1	4.4
	Watching Demos	Ours	84.4%	4.7	5.1
	Manually Crafted	Ours	95.6%	5.5	5.5



► Reasoning for Efficiency and Focus

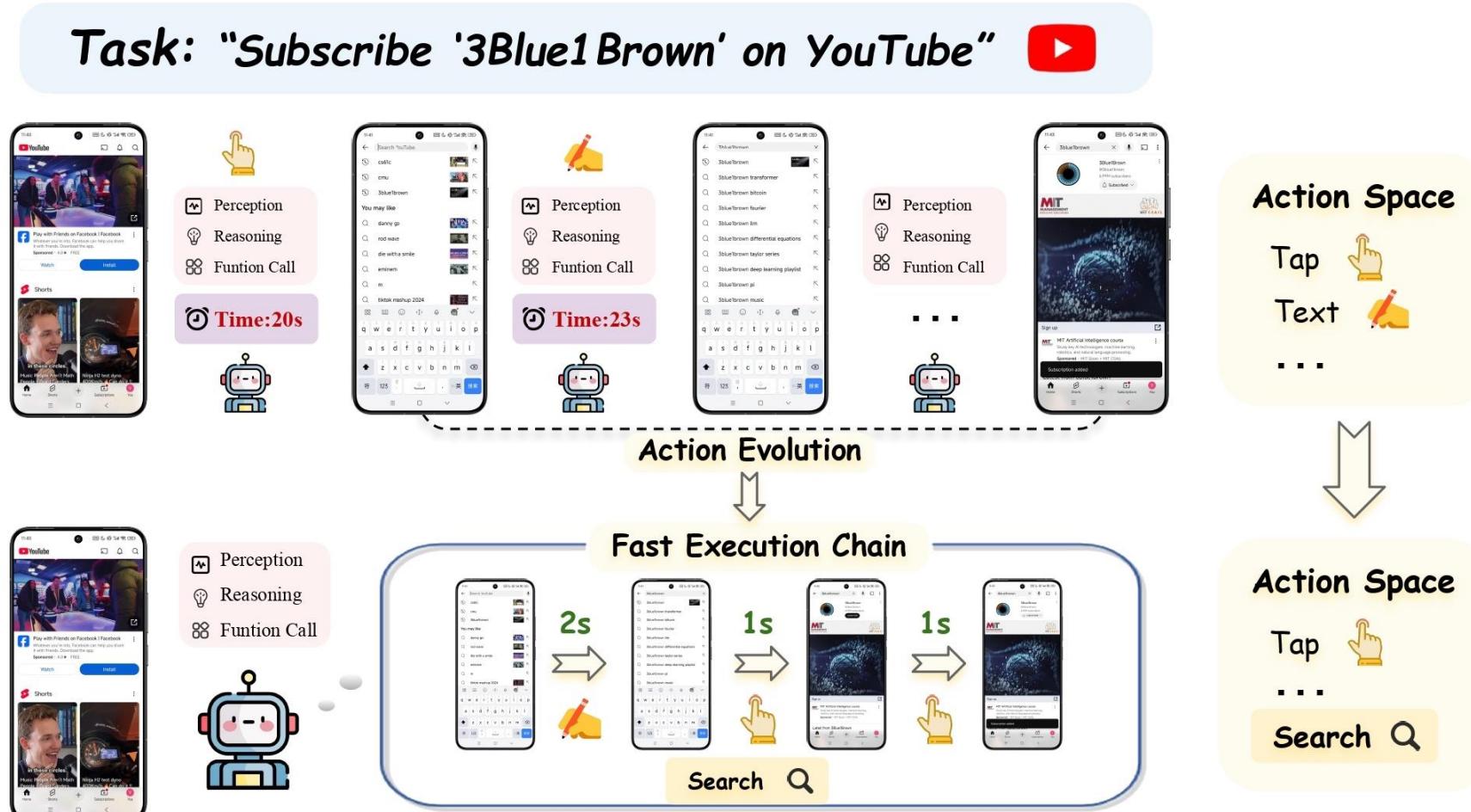
Task: "Subscribe '3Blue1Brown' on YouTube" 



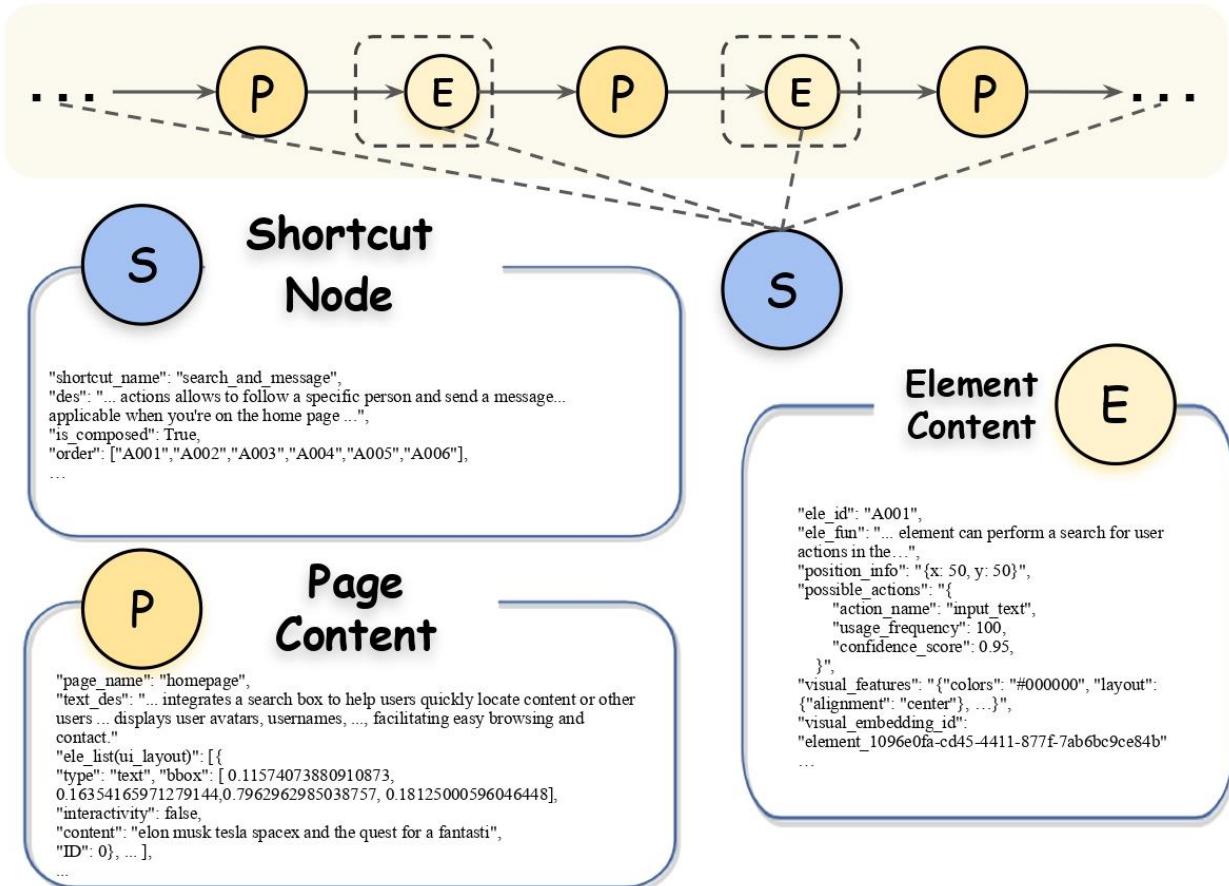
Action Space

Tap 
Text 

► Reasoning for Efficiency and Focus



► Trajectory as a chain

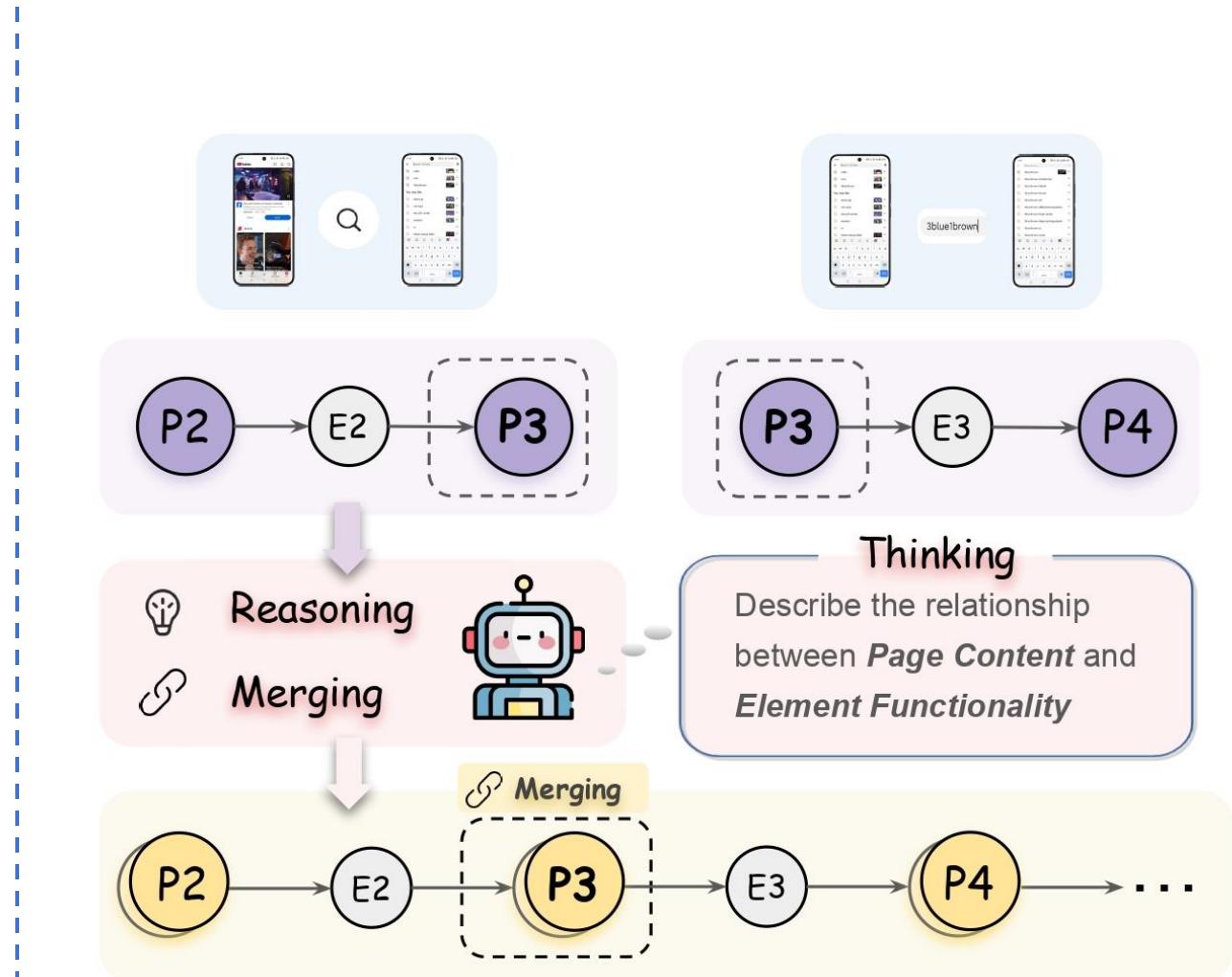


- The entire mobile phone operation process can be regarded as a **chain**, which includes **page nodes** and **element nodes**.
- Each node contains **all possible information**, such as function description, location, visual features, and so on.

► AppAgentX

How to make agent understand pages and elements?

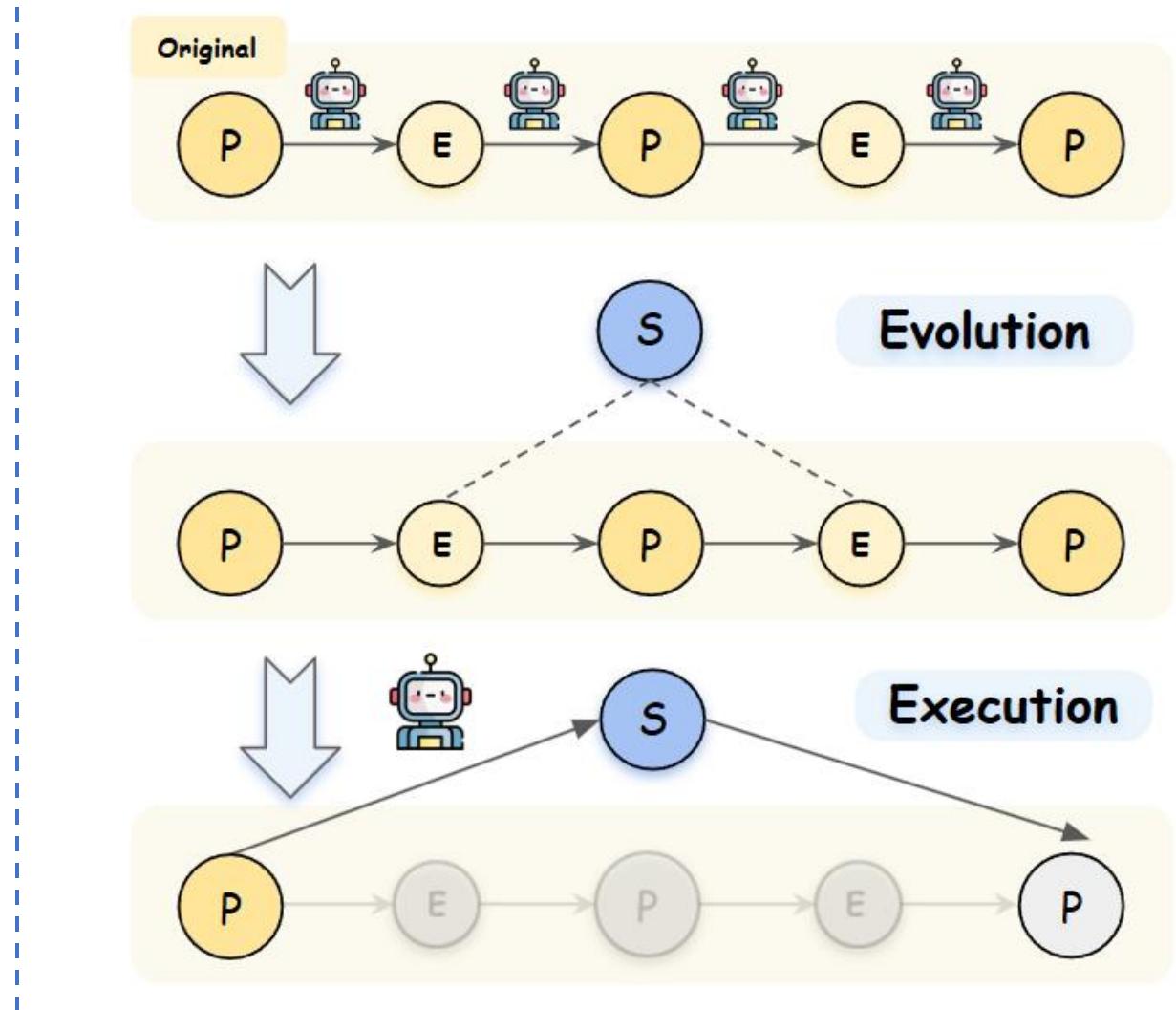
- The trajectory of a task execution is decomposed into multiple overlapping triples.
- Based on these triples, the LLM generates functional descriptions of both pages and UI elements.
- The descriptions of pages that are repeatedly generated are then merged. The entire interaction history is recorded using a chain of nodes.



► AppAgentX

Evolution Mechanism

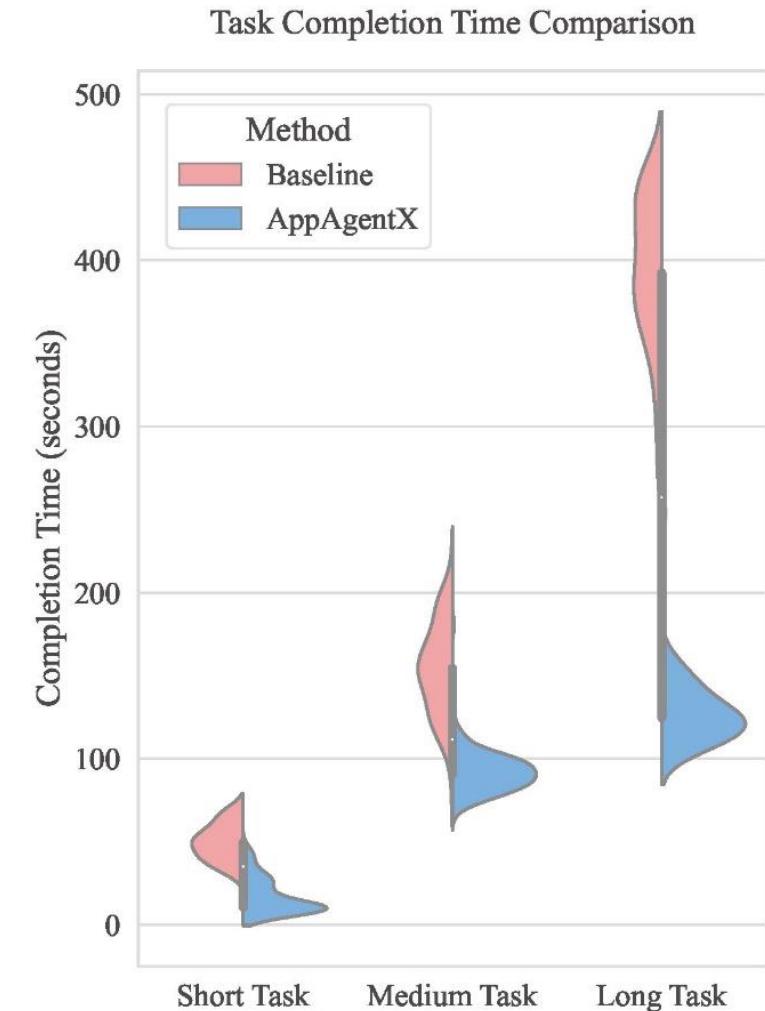
- The evolution mechanism identifies repetitive action sequences and creates high-level shortcuts, significantly reducing the number of steps and reasoning required for common tasks.



► AppAgentX

Method	Memory Type	Action Space	Steps↓	Step Time (s)↓	Tokens (k)↓	SR ↑
GPT-4o (Baseline)	None	Basic	10.8	26	6.72	16.9%
	Element	Basic	9.3	24	8.46	69.7%
AppAgentX	Chain	Basic	9.1	23	9.26	70.8%
	Chain	Basic+Evolve	5.7	16	4.94	71.4%

Table 1: **Analysis of Different Components in AppAgentX.** This table compares the performance differences resulting from the different designs with the baseline. Both our memory design and evolution mechanism can improve success rate and efficiency.



► Summary

- Language reasoning -> Action reasoning
- In AppAgent, the agent reasons at the UI element level
- In AppAgentX : the agent reasons over the entire trajectory

Thank you for your listening



Email: chizhang@westlake.edu.cn Homepage: <https://icoz69.github.io/>