Appendix B - AI Agentic Interactions: From GUI to Real World environment

AI agents are increasingly performing complex tasks by interacting with digital interfaces and the physical world. Their ability to perceive, process, and act within these varied environments is fundamentally transforming automation, human-computer interaction, and intelligent systems. This appendix explores how agents interact with computers and their environments, highlighting advancements and projects.

Interaction: Agents with Computers

The evolution of AI from conversational partners to active, task-oriented agents is being driven by Agent-Computer Interfaces (ACIs). These interfaces allow AI to interact directly with a computer's Graphical User Interface (GUI), enabling it to perceive and manipulate visual elements like icons and buttons just as a human would. This new method moves beyond the rigid, developer-dependent scripts of traditional automation that relied on APIs and system calls. By using the visual "front door" of software, AI can now automate complex digital tasks in a more flexible and powerful way, a process that involves several key stages:

- **Visual Perception:** The agent first captures a visual representation of the screen, essentially taking a screenshot.
- **GUI Element Recognition:** It then analyzes this image to distinguish between various GUI elements. It must learn to "see" the screen not as a mere collection of pixels, but as a structured layout with interactive components, discerning a clickable "Submit" button from a static banner image or an editable text field from a simple label.
- **Contextual Interpretation:** The ACI module, acting as a bridge between the visual data and the agent's core intelligence (often a Large Language Model or LLM), interprets these elements within the context of the task. It understands that a magnifying glass icon typically means "search" or that a series of radio buttons represents a choice. This module is crucial for enhancing the LLM's reasoning, allowing it to form a plan based on visual evidence.
- **Dynamic Action and Response:** The agent then programmatically controls the mouse and keyboard to execute its plan—clicking, typing, scrolling, and dragging. Critically, it must constantly monitor the screen for visual feedback,

dynamically responding to changes, loading screens, pop-up notifications, or errors to successfully navigate multi-step workflows.

This technology is no longer theoretical. Several leading AI labs have developed functional agents that demonstrate the power of GUI interaction:

ChatGPT Operator (OpenAI): Envisioned as a digital partner, ChatGPT Operator is designed to automate tasks across a wide range of applications directly from the desktop. It understands on-screen elements, enabling it to perform actions like transferring data from a spreadsheet into a customer relationship management (CRM) platform, booking a complex travel itinerary across airline and hotel websites, or filling out detailed online forms without needing specialized API access for each service. This makes it a universally adaptable tool aimed at boosting both personal and enterprise productivity by taking over repetitive digital chores.

Google Project Mariner: As a research prototype, Project Mariner operates as an agent within the Chrome browser (see Fig. 1). Its purpose is to understand a user's intent and autonomously carry out web-based tasks on their behalf. For example, a user could ask it to find three apartments for rent within a specific budget and neighborhood; Mariner would then navigate to real estate websites, apply the filters, browse the listings, and extract the relevant information into a document. This project represents Google's exploration into creating a truly helpful and "agentive" web experience where the browser actively works for the user.

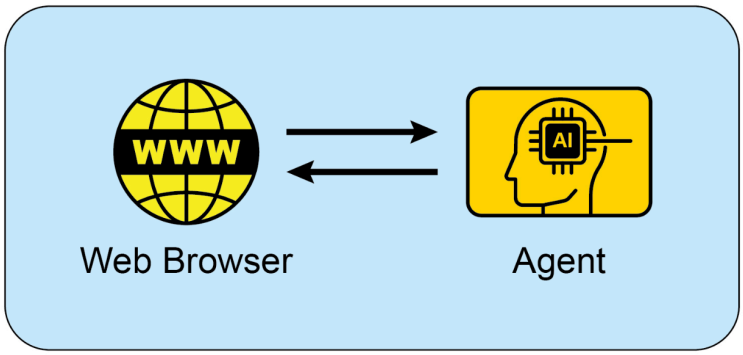

Fig.1: Interaction between and Agent and the Web Browser

Anthropic's Computer Use: This feature empowers Anthropic's AI model, Claude, to become a direct user of a computer's desktop environment. By capturing screenshots to perceive the screen and programmatically controlling the mouse and keyboard, Claude can orchestrate workflows that span multiple, unconnected applications. A user could ask it to analyze data in a PDF report, open a spreadsheet application to perform calculations on that data, generate a chart, and then paste that chart into an email draft—a sequence of tasks that previously required constant human input.

Browser Use: This is an open-source library that provides a high-level API for programmatic browser automation. It enables AI agents to interface with web pages by granting them access to and control over the Document Object Model (DOM). The API abstracts the intricate, low-level commands of browser control protocols, into a more simplified and intuitive set of functions. This allows an agent to perform complex sequences of actions, including data extraction from nested elements, form submissions, and automated navigation across multiple pages. As a result, the library facilitates the transformation of unstructured web data into a structured format that an AI agent can systematically process and utilize for analysis or decision-making.

Interaction: Agents with the Environment

Beyond the confines of a computer screen, AI agents are increasingly designed to interact with complex, dynamic environments, often mirroring the real world. This requires sophisticated perception, reasoning, and actuation capabilities.

Google's **Project Astra** is a prime example of an initiative pushing the boundaries of agent interaction with the environment. Astra aims to create a universal AI agent that is helpful in everyday life, leveraging multimodal inputs (sight, sound, voice) and outputs to understand and interact with the world contextually. This project focuses on rapid understanding, reasoning, and response, allowing the agent to "see" and "hear" its surroundings through cameras and microphones and engage in natural conversation while providing real-time assistance. Astra's vision is an agent that can seamlessly assist users with tasks ranging from finding lost items to debugging code, by understanding the environment it observes. This moves beyond simple voice commands to a truly embodied understanding of the user's immediate physical context.

Google's **Gemini Live**, transforms standard AI interactions into a fluid and dynamic conversation. Users can speak to the AI and receive responses in a natural-sounding voice with minimal delay, and can even interrupt or change topics mid-sentence, prompting the AI to adapt immediately. The interface expands beyond voice, allowing

users to incorporate visual information by using their phone's camera, sharing their screen, or uploading files for a more context-aware discussion. More advanced versions can even perceive a user's tone of voice and intelligently filter out irrelevant background noise to better understand the conversation. These capabilities combine to create rich interactions, such as receiving live instructions on a task by simply pointing a camera at it.

OpenAI's **GPT-4o model** is an alternative designed for "omni" interaction, meaning it can reason across voice, vision, and text. It processes these inputs with low latency that mirrors human response times, which allows for real-time conversations. For example, users can show the AI a live video feed to ask questions about what is happening, or use it for language translation. OpenAI provides developers with a "Realtime API" to build applications requiring low-latency, speech-to-speech interactions.

OpenAI's **ChatGPT Agent** represents a significant architectural advancement over its predecessors, featuring an integrated framework of new capabilities. Its design incorporates several key functional modalities: the capacity for autonomous navigation of the live internet for real-time data extraction, the ability to dynamically generate and execute computational code for tasks like data analysis, and the functionality to interface directly with third-party software applications. The synthesis of these functions allows the agent to orchestrate and complete complex, sequential workflows from a singular user directive. It can therefore autonomously manage entire processes, such as performing market analysis and generating a corresponding presentation, or planning logistical arrangements and executing the necessary transactions. In parallel with the launch, OpenAI has proactively addressed the emergent safety considerations inherent in such a system. An accompanying "System Card" delineates the potential operational hazards associated with an AI capable of performing actions online, acknowledging the new vectors for misuse. To mitigate these risks, the agent's architecture includes engineered safeguards, such as requiring explicit user authorization for certain classes of actions and deploying robust content filtering mechanisms. The company is now engaging its initial user base to further refine these safety protocols through a feedback-driven, iterative process.

Seeing AI, a complimentary mobile application from Microsoft, empowers individuals who are blind or have low vision by offering real-time narration of their surroundings. The app leverages artificial intelligence through the device's camera to identify and describe various elements, including objects, text, and even people. Its core functionalities encompass reading documents, recognizing currency, identifying

products through barcodes, and describing scenes and colors. By providing enhanced access to visual information, Seeing AI ultimately fosters greater independence for visually impaired users.

Anthropic's Claude 4 Series Anthropic's Claude 4 is another alternative with capabilities for advanced reasoning and analysis. Though historically focused on text, Claude 4 includes robust vision capabilities, allowing it to process information from images, charts, and documents. The model is suited for handling complex, multi-step tasks and providing detailed analysis. While the real-time conversational aspect is not its primary focus compared to other models, its underlying intelligence is designed for building highly capable AI agents.

Vibe Coding: Intuitive Development with AI

Beyond direct interaction with GUIs and the physical world, a new paradigm is emerging in how developers build software with AI: "vibe coding." This approach moves away from precise, step-by-step instructions and instead relies on a more intuitive, conversational, and iterative interaction between the developer and an AI coding assistant. The developer provides a high-level goal, a desired "vibe," or a general direction, and the AI generates code to match.

This process is characterized by:

- **Conversational Prompts:** Instead of writing detailed specifications, a developer might say, "Create a simple, modern-looking landing page for a new app," or, "Refactor this function to be more Pythonic and readable." The AI interprets the "vibe" of "modern" or "Pythonic" and generates the corresponding code.
- **Iterative Refinement:** The initial output from the AI is often a starting point. The developer then provides feedback in natural language, such as, "That's a good start, but can you make the buttons blue?" or, "Add some error handling to that." This back-and-forth continues until the code meets the developer's expectations.
- **Creative Partnership:** In vibe coding, the AI acts as a creative partner, suggesting ideas and solutions that the developer may not have considered. This can accelerate the development process and lead to more innovative outcomes.
- **Focus on "What" not "How":** The developer focuses on the desired outcome (the "what") and leaves the implementation details (the "how") to the AI. This

allows for rapid prototyping and exploration of different approaches without getting bogged down in boilerplate code.
- **Optional Memory Banks:** To maintain context across longer interactions, developers can use "memory banks" to store key information, preferences, or constraints. For example, a developer might save a specific coding style or a set of project requirements to the AI's memory, ensuring that future code generations remain consistent with the established "vibe" without needing to repeat the instructions.

Vibe coding is becoming increasingly popular with the rise of powerful AI models like GPT-4, Claude, and Gemini, which are integrated into development environments. These tools are not just auto-completing code; they are actively participating in the creative process of software development, making it more accessible and efficient. This new way of working is changing the nature of software engineering, emphasizing creativity and high-level thinking over rote memorization of syntax and APIs.

Key takeaways

- AI agents are evolving from simple automation to visually controlling software through graphical user interfaces, much like a human would.
- The next frontier is real-world interaction, with projects like Google's Astra using cameras and microphones to see, hear, and understand their physical surroundings.
- Leading technology companies are converging these digital and physical capabilities to create universal AI assistants that operate seamlessly across both domains.
- This shift is creating a new class of proactive, context-aware AI companions capable of assisting with a vast range of tasks in users' daily lives.

Conclusion

Agents are undergoing a significant transformation, moving from basic automation to sophisticated interaction with both digital and physical environments. By leveraging visual perception to operate Graphical User Interfaces, these agents can now manipulate software just as a human would, bypassing the need for traditional APIs. Major technology labs are pioneering this space with agents capable of automating complex, multi-application workflows directly on a user's desktop. Simultaneously, the next frontier is expanding into the physical world, with initiatives like Google's Project Astra using cameras and microphones to contextually engage with their surroundings.

These advanced systems are designed for multimodal, real-time understanding that mirrors human interaction.

The ultimate vision is a convergence of these digital and physical capabilities, creating universal AI assistants that operate seamlessly across all of a user's environments. This evolution is also reshaping software creation itself through "vibe coding," a more intuitive and conversational partnership between developers and AI. This new method prioritizes high-level goals and creative intent, allowing developers to focus on the desired outcome rather than implementation details. This shift accelerates development and fosters innovation by treating AI as a creative partner. Ultimately, these advancements are paving the way for a new era of proactive, context-aware AI companions capable of assisting with a vast array of tasks in our daily lives.

References

1. Open AI Operator, https://openai.com/index/introducing-operator/
2. Open AI ChatGPT Agent: https://openai.com/index/introducing-chatgpt-agent/
3. Browser Use: https://docs.browser-use.com/introduction
4. Project Mariner, https://deepmind.google/models/project-mariner/
5. Anthropic Computer use: https://docs.anthropic.com/en/docs/build-with-claude/computer-use
6. Project Astra, https://deepmind.google/models/project-astra/
7. Gemini Live, https://gemini.google/overview/gemini-live/?hl=en
8. OpenAI's GPT-4, https://openai.com/index/gpt-4-research/
9. Claude 4, https://www.anthropic.com/news/claude-4