



GestureGPT: Toward Zero-Shot Free-Form Hand Gesture Understanding with Large Language Model Agents

XIN ZENG*, Institute of Computing Technology, Chinese Academy of Sciences, China and University of Chinese Academy of Sciences, China

XIAOYU WANG*†, The Hong Kong University of Science and Technology, China

TENGXIANG ZHANG‡, Goertek Inc., China

CHUN YU, Computer Science and Technology, Tsinghua University, China

SHENGDONG ZHAO, Synteraction Lab, City University of Hong Kong, China

YIJIANG CHEN‡, Institute of Computing Technology, Chinese Academy of Sciences, China and University of Chinese Academy of Sciences, China

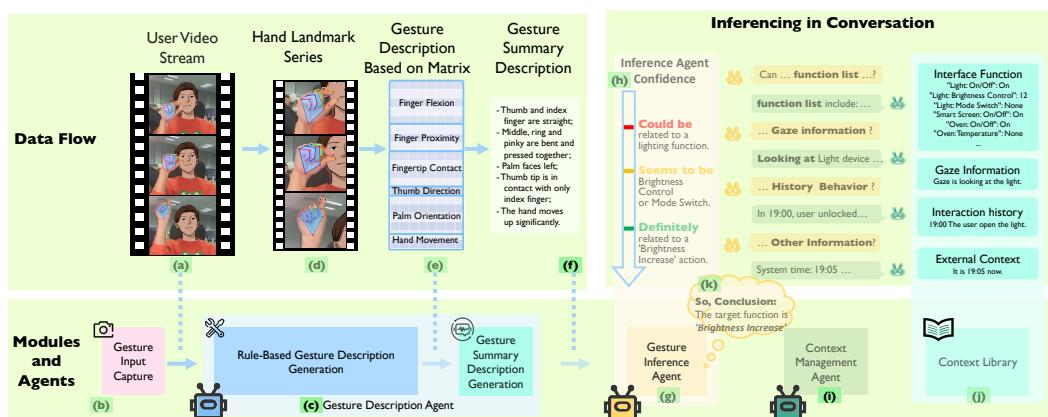


Fig. 1. This scenario illustrates a system implemented based on GestureGPT.

Existing gesture interfaces only work with a fixed set of gestures defined either by interface designers or by users themselves, which introduces learning or demonstration efforts that diminish their naturalness. Humans, on the other hand, understand free-form gestures by synthesizing the gesture, context, experience, and

*Both authors contributed equally to this research.

†This work was completed during an internship at ICT.

‡Corresponding author.

Authors' Contact Information: Xin Zeng, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China and University of Chinese Academy of Sciences, Beijing, China, zengxin18z@ict.ac.cn; Xiaoyu Wang, The Hong Kong University of Science and Technology, Hong Kong, China, xwangj@connect.ust.hk; Tengxiang Zhang, Goertek Inc., Beijing, China, ztxseuthu@gmail.com; Chun Yu, Computer Science and Technology, Tsinghua University, Beijing, China; Shengdong Zhao, Synteraction Lab, City University of Hong Kong, Hong Kong, China; Yiqiang Chen, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China and University of Chinese Academy of Sciences, Beijing, China, yqchen@ict.ac.cn.



This work is licensed under a Creative Commons Attribution 4.0 International License.

© 2024 Copyright held by the owner/author(s).

ACM 2573-0142/2024/12-ART545

<https://doi.org/10.1145/3698145>

common sense. In this way, the user does not need to learn, demonstrate, or associate gestures. We introduce GestureGPT, a free-form hand gesture understanding framework that mimics human gesture understanding procedures to enable a natural free-form gestural interface. Our framework leverages multiple Large Language Model agents to manage and synthesize gesture and context information, then infers the interaction intent by associating the gesture with an interface function. More specifically, our triple-agent framework includes a Gesture Description Agent that automatically segments and formulates natural language descriptions of hand poses and movements based on hand landmark coordinates. The description is deciphered by a Gesture Inference Agent through self-reasoning and querying about the interaction context (e.g., interaction history, gaze data), which is managed by a Context Management Agent. Following iterative exchanges, the Gesture Inference Agent discerns the user's intent by grounding it to an interactive function. We validated our framework offline under two real-world scenarios: smart home control and online video streaming. The average zero-shot Top-1/Top-5 grounding accuracies are 44.79%/83.59% for smart home tasks and 37.50%/73.44% for video streaming tasks. We also provide an extensive discussion that includes rationale for model selection, generalizability, and future research directions for a practical system *etc.*

CCS Concepts: • **Human-centered computing** → **Gestural input; User interface management systems**.

Additional Key Words and Phrases: Free-Form Gesture, Zero-Shot, Gesture Recognition, Interaction Context

ACM Reference Format:

Xin Zeng, Xiaoyu Wang, Tengxiang Zhang, Chun Yu, Shengdong Zhao, and Yiqiang Chen. 2024. GestureGPT: Toward Zero-Shot Free-Form Hand Gesture Understanding with Large Language Model Agents. *Proc. ACM Hum.-Comput. Interact.* 8, ISS, Article 545 (December 2024), 38 pages. <https://doi.org/10.1145/3698145>

1 Introduction

Gestures express human intent intuitively and promptly, enabling natural human-computer interaction with low cognitive load [50, 72]. Most existing gesture interfaces support only predefined gestures, which can achieve a very high gesture classification accuracy [3, 65]. However, it requires significant effort to learn and memorize different gestures and their respective mappings to system functions, particularly with a large set of gestures [11, 15]. The gesture-function mapping is also fixed, which cannot be easily expanded and may contradict users' preferences [8, 45, 64].

To address the limitations of predefined gestures, researchers have proposed gestural interfaces that support user-defined gestures [37, 67, 73]. Users can define their own gestures for each function with only a few demonstrations, thus eliminating the learning effort and enhancing system flexibility. However, each user must design and demonstrate their own gesture and associate it with a system function [41, 57]. Users also still need to memorize their own gestures, which may lead to frustration using the interface, especially when there are a large number of functions. Such constraints degrade the natural interaction experience and hinder the broad adoption of gestural interfaces, leading to Norman's famous argument that "Natural User Interfaces are Not Natural" [46]. While significant advances have been made in gesture recognition technologies over the years, this naturalness challenge of gestural interfaces persists. To address the pursuit of more natural gestural expressions, users expect high-frequency gestures to seamlessly link various functions (e.g., common touchscreen gestures like zooming in/out, or typical semantic gestures like thumbs-up and rock-and-roll) without being confined to a limited set of expressions or frequently needing to customize their own [26, 34]. Additionally, as gestural interfaces become more widespread, users will likely experience significant improvements in both their familiarity with these expressions and the speed of their interactions [44, 58].

To that end, future gestural interfaces require direct, end-to-end **gesture understanding** rather than merely **gesture recognition**. Users should not have to learn, memorize, or demonstrate specific gestures. Instead, they can perform gestures naturally, according to their understanding of the function's semantics and their everyday interaction experiences with humans and machines.

The interface automatically links the gesture to its corresponding function, considering both the gesture and the interaction context. This gesture understanding task is inherently more challenging than simple gesture recognition.

We propose GestureGPT, an LLM-based paradigm that comprehends natural free-form hand gestures and automatically links them to their intended functions. The core idea of GestureGPT is to utilize the LLM's rich common sense for recognizing gestures and understanding the interaction context, as well as its potent inference capabilities to map gestures to their intended functions. Constructing such an LLM-based gesture understanding framework demands considerable effort. For instance, gestures and context information must be transformed and formatted in a way that LLMs can process. Additionally, there needs to be a mechanism for robust and thorough synthesis of all relevant information without overlooking any details.

To that end, we design a triple-agent paradigm that generates gesture descriptions, manages the context, and infers the intended function. This involves a *Gesture Description Agent*, a *Gesture Inference Agent*, and a *Context Management Agent*, all based on LLMs. 1) The *Gesture Description Agent* describes hand gestures in natural languages to facilitate the powerful text-based LLMs' reading and understanding of gestures. A set of specially designed and tuned rules transforms hand poses and movements into discrete states (e.g., the bending state of each finger) based on visually extracted hand landmarks, triggered by raising the hand to the chest level. The agent then synthesizes a general description of the gesture based on a matrix formed with the states. 2) The *Gesture Inference Agent* analyzes the description and initiates a conversation with the *Context Management Agent* to inquire about the context information. After multiple rounds of dialogue, it infers which interactive function the user intends to activate. 3) The *Context Management Agent* answers questions from the *Gesture Inference Agent* by referring to a context library, which includes various types of context information, such as gaze points and interaction history, etc. The context library is organized in JSON file format.

Figure 1 envisions a process in which a system, implemented through the GestureGPT concept, understands a user's gesture. In this scenario, he (a) triggers the system with a raised right hand above the chest, followed by looking at the light and performing a "pinch and move up" gesture. The (b) input camera captures this action through real-time video streaming. (c) The *Gesture Description Agent* processes the gesture by (d) filtering key frames from the video and extracting hand landmarks to (e) generate the gesture state matrix. Subsequently, (f) *Gesture Description Agent* generates a description of the gesture's pose and movement. Upon activation by the description, the (g) *Gesture Inference Agent* analyzes this description to determine which function the gesture maps to and (h) assesses the confidence level of the result. If the confidence is deemed insufficient, *Gesture Inference Agent* requests context information from the (i) *Context Management Agent*, which retrieves relevant data from the (j) context library containing all available context information in the current environment, to inform the inquiry. According to the inquiry, as context information such as "Gaze: user is looking at the light" and "History: the user turned on the light at 19:00" is incorporated, the *Gesture Inference Agent* (k) concludes that the function designated by the gesture is to "increase the brightness" of the light. The function could then be triggered to finish the interaction process.

GestureGPT offers a number of advantages that address the challenges and limitations of previous gestural interfaces. 1) GestureGPT allows users to perform natural free-form hand gestures, which do not need to resemble those in a predefined gesture set or those previously defined by users. This eliminates the need for learning, memorizing, or demonstrating specific gestures. 2) GestureGPT automatically associates gestures with their corresponding interface functions through a step-by-step inference process based on both the gesture description and the interaction context. Such a novel LLM-based structure successfully addresses the ephemeral nature of gestures and their close

relationship with context [46]. Additionally, the integration of context further enhances the accuracy of gesture mapping. 3) GestureGPT analyzes spatial coordinates of hand landmarks to generate gesture descriptions, making it independent of view angles and even modalities. For example, hand landmarks can also be reconstructed by wearable sensors [28, 81]. Such flexibility makes it easy to adapt GestureGPT for a wide range of applications. The natural language descriptions also preserve user privacy and reduce data transmission load, which are important for interaction interfaces.

Describing nuanced finger states and movements is crucial for accurately understanding hand gestures. Our hand landmarks-based method captures finger states accurately and thoroughly by first calculating a set of discrete finger states based on rules, then using an LLM to synthesize a gesture summary. The rules' parameters are tuned using third-person view public gesture image datasets and tested on both third-person view (HaGRID [30], 38576 test samples) and first-person view datasets (EgoGesture [80], approximately 5000 test samples). The error rates are 2.3% for third-person views and 6.3% for first-person views, respectively. Two gesture experts rated the synthesized summary, which achieved a score of 3.51 (std = 1.14) on a 5-point Likert scale, where 1 indicates that almost none of the description is correct and relevant, and 5 indicates that almost all of it is.

To evaluate our framework, we conducted experiments under two realistic interaction scenarios: smart home IoT device control using an AR headset (first-person view with 18 functions) and online video streaming on a desktop PC (third-person view with 66 functions). The highest zero-shot Top-1/Top-5 gesture grounding accuracies achieved were 44.79%/83.59% for smart home control and 37.50%/73.44% for video streaming. We report accuracies at various levels to more comprehensively understand and demonstrate the framework's performance and boundaries. The results demonstrate the significant potential of GestureGPT, which, to the best of our knowledge, is **the first zero-shot free-form gesture understanding framework** that requires no learning, memorization, demonstration, or association efforts.

However, GestureGPT is not yet a practical interface ready for everyday use, primarily due to the slow inference speeds of current LLM systems. Our current system averages 227 seconds per task in our evaluations, a delay caused by long prompts, multi-turn dialogues, and rate limits enforced by LLM cloud services. Instead, it introduces a new **paradigm** for gestural interfaces. To the best of our knowledge, GestureGPT is the first feasible framework for the inherently difficult task of understanding free-form gestures. Previous efforts in gesture recognition have not successfully addressed free-form gestures, let alone the more complex challenge of gesture understanding. GestureGPT, therefore, not only pioneers a new approach but also lays the foundational framework for future advancements. This framework can also be easily expanded to recognize a broader range of user intents through different modalities, such as facial expressions and body postures, among others. Technologies such as edge-side LLMs or specifically fine-tuned large multi-modal models could significantly reduce response times, paving the way for a more practical implementation of this paradigm in the future.

Our contribution is three-fold:

- (1) We proposed and evaluated the first framework that mimics human reasoning processes to achieve automatic free-form hand gesture understanding, as to our best knowledge.
- (2) We designed a set of gesture description rules based on hand landmarks to thoroughly and accurately capture states and movements of the hand, which has comparable performance to SOTA large multi-modal GPT-4o.
- (3) We carefully crafted prompts for each agent, analyzed the offline evaluation results, and generated various insights. Such insights are invaluable for future context-aware agent-based gesture understanding work.

2 Background and Related Work

2.1 LLM as Autonomous Agent

Large language models have displayed an exceptional ability to understand and execute a broad spectrum of tasks [2, 47]. LLM has the potential to emulate human-level intelligence, accurately perceive generalized environments, act accordingly, and iterate to enhance outcome [55] when faced with diverse situations [66]. Agents based on LLMs have shown potential in various domains, from web browsing [9, 74, 83], strategic planning [75] to robotic control [4, 10]. This paper, on the other hand, addresses a notable gap in existing literature and utilizes LLMs for free-form gesture understanding and interaction.

Though LLM agents have shown promising intelligence, a single agent implementation may suffer from performance degradation in long context scenarios [27]. Instead, multi-agent systems have shown superiority to accomplish more complex tasks in collaboration, reducing hallucination [68], and information exchange [61]. For instance, Park et al. [49] designed a multi-agent system that simulates human behavior in a virtual environment. Park et al. [48] relies on conversational interactions among multiple agents to aid online decision-making, which shows multi-agent's great potential in reasoning under unfamiliar scenarios. Other applications software development [51], span reasoning [35], evaluation [5, 79], and a myriad of intricate tasks [48, 85]. So, GestureGPT chooses a triple-agent architecture to better handle the complicated context-aware gesture understanding task. The goals of the three agents are clearly defined and isolated, so that they can be optimized individually while collaborate seamlessly to achieve accurate gesture comprehension.

2.2 Natural Free-form Gesture Understanding

Gestural interfaces working with pre-defined gesture set demand large annotated dataset, and the confinement to pre-defined gestures also hampers the naturalness of interaction [70]. The conflicts in different system designs further exacerbate user adaptation challenges across platforms. User-defined gestural interfaces mitigates the learning burden by allowing users to define their own gestures. Only several demonstrations of the gesture are necessary for the system to learn new gestures with the help of advanced few-shot learning algorithms [73]. Gesture Coder [37] allows the user to demonstrate a gesture, and, instead of defining a gesture name for it, the user directly associate it to a designated function with auto-generated recognizer. But in these works, users still need to demonstrate and memorize the gesture while ensuring its distinctiveness from existing gesture commands.

Nevertheless, all aforementioned approaches are still limited by a finite set of distinguishable gestures defined by interface designers or users. This leads to a mismatch between the fixed gesture set and the flexible gestures humans perform in different scenarios, greatly restricting gesture expressiveness. Thus, both pre-defined and user-defined gestural interfaces require users to adapt to them, rather than the reverse [70]. Free-form gestural interfaces, however, do not have these limitations. For instance, Gesture Avatar [36] enables any drawing gesture on a screen. The input gesture is linked to a UI element based on appearance resemblance, providing an intuitive interaction experience. Yet, gaps remain in the research of free-form hand gesture understanding. Compared with screen gestures, hand gestures are more complex, do not necessarily resemble UI elements, and can vary under different scenarios for the same function.

Recent advances in zero-shot learning make it possible to recognize unseen gesture classes [39, 40, 71]. Madapana [39, 40] proposed frameworks that define a set of attributes for gestures, recognizing unseen gestures by manually labeling these attributes for each unseen gesture. Similarly, Wu et al. [71] introduced a prototype-based approach that requires the definition of “semantic prototypes”

for unseen gesture categories. These methods explore to some extent the recognition of “unseen” gestures, enabling the system to recognize new gestures without additional training.

However, above methods focus solely on the task of gesture recognition. Even for the recognition task, they only identify key features of new gestures, not their semantic names. Gesture understanding, however, requires not only recognizing gestures but also mapping them to functions, which demands a comprehensive integration of complex contextual information (such as available functions, interaction history, and physical environment). Such analysis and reasoning necessitate mimicking human thought processes, a task that traditional machine learning models struggle to achieve.

To tackle such challenges, our framework relies on LLMs to understand the semantic meanings of both the hand gestures and the system functions to ensure correct gesture-function mapping, because LLMs are currently the most promising tools capable of human-alike inference and comprehensive context understanding. An overview of current gesture-based interaction systems and the placement of our work is shown in Figure 2.

	Related Work		User Efforts			
	References	Gesture Type	Learning When using the system for the first time	Memorization When using the system later on	Demonstration For recording the gesture example	Association For mapping the input to a function
Pre-defined Gestures	Montero and Sucar, 2006 Tarrana li et al., 2015 Madipana and Wachs, 2018 Wu et al., 2021	Hand Gesture / Screen Gesture / Body Gesture / -	+	+	-	-
User-defined Gestures	Gesture Coder (Li and Li, 2012) Xu et al., 2022	Hand Gesture / Screen Gesture	-	+	+	+
Free-form Gestures	Gesture Avatar (Li and Li, 2011)	Screen Gesture	-	-	-	+
	Wavelblat, 1993 ⁶	Hand Gesture	-	-	-	-
GestureGPT		Hand Gesture	-	-	-	-

Fig. 2. Overview of current gestural interaction systems.

2.3 Gesture Understanding with Context Information

As Norman pointed out, gestures are ephemeral in nature and highly context related [46]. They inherently possess diverse semantics across different contexts, and may embody social metaphors [59]. Contextual information such as spatial distance [42], gaze [6], speech [70], user history [7, 43] and domain-specific knowledge [62] have been integrated into gestural interaction systems to improve gesture recognition accuracy. However, most previous work only leverage gaze information for simplified tasks [1, 16, 23, 54, 56]. The handling of different types of context typically require highly specialized models [7, 24, 56], which limits the generalizability of such systems.

LLM agents’ promising ability to solve complex problems provides an alternative solution for inference based on different types of context information. LLM agents can retrieve higher-level context like semantic meaning of interaction elements [21] and users’ profile and preferences [52]. The way it utilizes context and the intentions it can deal with are also greatly enlarged. For example, [20, 31, 32, 52] have shown that LLM agents can turn loosely-constrained commands into appropriate actions. Inspired by existing research, we use a *Context Management Agent* to manage context information stored in the context library, and a *Gesture Inference Agent* to infer the intention behind a gesture based on context cues.

3 Method

GestureGPT has a triple-agent framework to efficiently manage gesture description, context, and inference tasks, enhancing system flexibility and scalability while addressing the shortcomings of single or dual agent systems, consists of: (1) **Gesture Description Agent**, generating descriptions of gestures from videos; (2) **Context Management Agent**, handling interaction context; and (3)

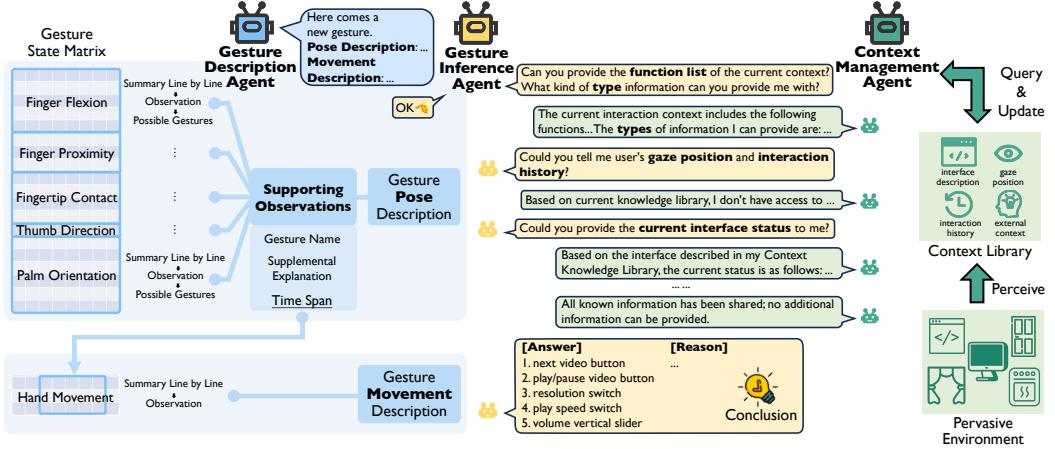


Fig. 3. Agents Collaboration Workflow.

Gesture Inference Agent, synthesize information to infer gesture intentions through dialogue and reasoning.

The workflow initiates with the transformation of user gestures, captured by an RGB camera, into natural language description. The *Gesture Description Agent* first uses a set of rules to transform the gesture into a “Gesture State Matrix”, which delineates hand and finger states over time. The agent then relies on this matrix to produce a summary of gestures, which is forwarded to the *Gesture Inference Agent*. Upon receiving the gesture description, the *Gesture Inference Agent* engages in multi-round dialogue with the *Context Management Agent*. This involves identifying and soliciting relevant context information from a context library managed by the *Context Management Agent*. Through iterative dialogue, the *Gesture Inference Agent* gains a comprehensive understanding of the interactive scenario and completes a dynamic mapping between the gesture and possible functions.

3.1 Gesture Description Agent

The Gesture Description Agent is crucial for translating video-captured gestures into natural language descriptions that is understandable by LLMs. We choose LLM since it has richer common knowledge and stronger inference than existing Large Multimodal Models (LMMs) that we can access, which is vital to deal with context-aware free-form gesture understanding tasks.

3.1.1 Rule-Based Gesture Description Generation We design a set of 6 rules to encode the hand gesture in terms of finger flexion, proximity, contact, direction, as well as palm orientation and hand position (Table 1). The detailed rule definition can be found in Appendix A.1.

Each state is calculated using the coordinates of hand landmarks, as shown in Fig 4, which are generated by MediaPipe [77] from videos captured with an RGB camera. For each rule, parameters (such as the distance between fingers to determine their proximity) are trained on a third-person view (*i.e.*, viewed from an external perspective, typically through a camera not aligned with the subject’s viewpoint) public gesture dataset, and tested on first-person view (*i.e.*, viewed from the subject’s own perspective, typically through a head-mounted camera) and third-person view datasets. Test results show an error rate of 2.3% for third-person view and 6.3% for first-person view samples, indicating the rules’ efficacy in retaining accurate information from gestures across different viewpoints, validating it a universal solution applicable to various gesture capture devices.

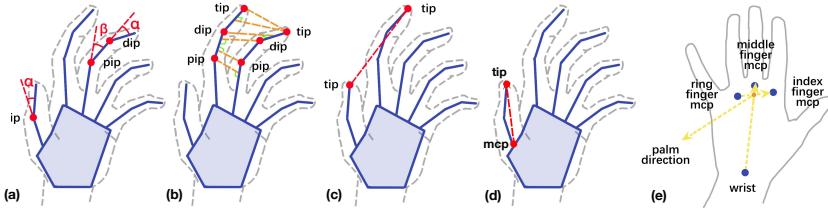


Fig. 4. Illustration of gesture description rules. (a) The flexion of a finger is calculated from the sum of bending angle of ip joint (for thumb) or pip and dip joint (for other fingers). (b) The proximity of two fingers is calculated from the average distance from each finger’s pip/dip/tip joint to the other finger. (c) The contact of thumb and another finger is calculated from the distance of their fingertips. (d) The pointing direction of thumb is calculated from the vector from thumb’s mcp to tip. (e) The palm orientation is calculated from the dot product of the two vectors on the hand, pointing towards the reader.

The details of parameter training and the evaluation metrics for these rules are thoroughly described in Appendix A. The pseudocode for the rules is provided in Appendix A.3. An evaluation and error analysis of the rules can be found in Appendix A.2.

The gesture period of interest starts when users raises their hands to or above their chest level. Frames within a gesture period are processed at 0.2-second intervals, with each sampled frame undergoing rule-based calculations. However, the natural language descriptions concatenated from each rule are too long to fit into a prompt of an LLM. So we introduce a gesture state matrix that contains vectors of each frame, which capitalizes on LLMs’ proficiency with code-formatted content [14]. The detailed definition of the gesture state matrix is given in Appendix A.4.

3.1.2 Gesture Summary Description Generation The system needs to extract the rich information embedded in the matrix and generate a concise summary of gesture descriptions for the *Gesture Inference Agent*. A significant challenge is the gesture state matrix data type; the pre-trained data of the LLM likely does not contain exactly similar data. We employ a chain-of-thought [69] process to guide the LLM step-by-step, supplemented by high-quality expert examples that demonstrate how the analysis should be conducted. We choose to use two prompts to generate the hand **pose** (finger flexion, finger proximity, fingertip contact, thumb direction and palm orientation) and hand **movement** description separately. As is shown in Figure 3, the Gesture Description Agent first converts pose-related rows of the matrix to descriptions of possible gestures, along with the time span during which this gesture happened. The time span is then used to select the corresponding part of movement-related rows, and another prompt is used to convert it to movement description. As long as the agent correctly finds the valid pose of the gesture, irrelevant movements are naturally filtered out.

For pose description, we developed a prompt structured into three parts shown in Figure 5: Introduction, Procedure and Examples. The introduction outlines the task, describes the the gesture state matrix, and explains the concept of “interactive gestures” versus non-interactive hand movements, helping the agent to distinguish between different types of gesutres and understand their lifecycle. The procedure section teaches the agent to understand the gesture state matrix and guess possible gestures from the matrix, while addressing some common mistakes to regulate its behaviors. In paticular, it instructs the agent to decompose the matrix, guess possible gestures from each part, and synthesize the conclusions from all parts to get the most possible gestures, encouraging a human-like process of understanding and summarizing data, while using transcription to counter LLM’s forgetfulness and improve accuracy. The last part provides two typical examples of static and dynamic gestures to enhance LLM performance by reflecting back to the guidelines stated before. This well-structured prompt ensures that the LLM model correctly processes the matrix data type.

Table 1. Summary of rules, their meanings, and corresponding values.

Rule Name	Applicable to	Calculation	Value
Finger Flexion	Thumb, Index, Middle, Ring, Pinky	measured as the total bending angle of each joint: for the thumb, the IP joint; for other fingers, the PIP and DIP joints.	1: Straight; 0: Between; -1: Bent
Finger Proximity	Index-Middle, Middle-Ring, Ring-Pinky	calculated as the average minimal distance from each finger's joint to the other finger.	1: Pressed Together; 0: Between; -1: Apart
Thumb Fingertip Contact	Thumb-Index, Thumb-Middle, Thumb-Ring, Thumb-Pinky	computed as the distance between their fingertips.	1: Contact; 0: Between; -1: No Contact
Pointing Direction	Thumb	the direction from thumb's mcp joint to tip joint.	1: Upward; -1: Downward; 0: Other Directions/Bent
Palm Orientation	Palm	computed as the direction to which the palm is facing.	One-hot encoding: [Left, Right, Down, Up, Inward, Outward]; All zeros: Unknown
Hand Position	Hand	computed as the geometrical center of a hand by taking average of all 21 landmarks' coordinates.	Float coordinates

The hand movement description prompt shares a similar structure. The generated description of the gesture will be like:

- Thumb and Index Finger transition from non-specific/bent to straight.
- Middle, Ring, and Pinky Fingers transition from bent to straight.
- Fingers start close together and then spread apart.
- Thumb starts in contact with all fingertips but then moves away.
- Palm orientation starts facing left but becomes non-specific.
- The hand moves left slightly with negligible vertical and depth movements.

3.2 Gesture Inference Agent

Upon receiving gesture descriptions from Gesture Description Agent, the Gesture Inference Agent engages in a dialogue with Context Management Agent. This conversational exchange is pivotal for the Gesture Inference Agent to discern the user's actual intent in the context, *i.e.*, associating the gesture with a target system function from a list of potential functions supplied by Context

Gesture Description Agent - Pose (Prompt Overview)	Gesture Inference Agent (Prompt Overview)
<pre> # Introduction: ## Task Introduction: Your task is to analyze hand poses ... identifying the gesture ... ## Input Data Introduction: A 2D array ... represents hand and finger states. Each time step is 0.2 seconds ... ## Gesture Introduction: Users can use static or dynamic gestures to interact ... # Procedure: ## step 1: Decompose the input array ... ## step 2: Synthesize observations and guesses gesture ... ## Behavior Guide: things should and should not do ... Common Mistakes ... # Examples # Example 1 (A specific example of <u>Static</u> Gesture 'peace') # Example 2 (A specific example of <u>Dynamic</u> Gesture 'Zoom In with Two Fingers') </pre>	<pre> # Introduction: ## Task Introduction: you will understand an interactive gesture from its description map it with a specific function ... ## Gesture Description Introduction: The init input contains content related to hand posed and hand movements... ## Context Management Agent Introduction: The **Context Agent** manages all type of interaction context information... # Behavior Guidelines: ## Requirements: Things need to do like 'asking for the function list' and 'requesting one context at a time'... ## Prohibitions: Things can not do like 'ask **Context Agent** to align the gesture to a function' ... ## Output Block: The output includes 'Thought', 'Gesture Guess', and 'Question for Context Agent' when the information is insufficient. It contains 'Thought' and 'Results' once a decision has been made. All outputs are requested in JSON format. </pre>

Fig. 5. Prompt for Gesture Description Agent. Fig. 6. Prompt for Gesture Inference Agent.

Management Agent. This setup epitomizes a collaborative conversation aimed at overcoming the challenges of gesture ambiguities under different interactive scenarios and context.

We outline the agent’s tasks and some behavioral guidelines to better demonstrate the semantic nature of the agent task revolves around thinking, summarizing, and inferring. The overall prompt is structured into two parts shown in Figure 6.

The introduction part includes a Task Introduction, a Gesture Description Introduction, and a Context Management Agent Introduction (its conversational counterpart). A critical aspect to convey in this part is that the initial gesture description may contain errors. Therefore, the Gesture Inference Agent must utilize context information to help with the inference, correct potential mistakes, and make decisions. Behavioral guidelines for the agent consist of seven directives and five prohibitions, acquired by iteratively updating the prompts during implementation. When new error tendencies are observed, corrections are introduced into the prompt, akin to a teacher’s guidance. Then, we define the output format. We opt for JSON due to its ease of parsing, allowing for efficient extraction of useful information for subsequent conversational rounds. Importantly, we require the agent to first articulate its **thoughts** before posing **questions** or drawing **conclusions**. This approach has been proven to make the agent’s behavior more reasonable. Once the agent deems that it is confident to decide or that no further context can be gleaned, it proceeds to make a final decision, listing the top-5 possible functions for the current interface, ranked from most to least likely. The Gesture Inference Agent emphasizes the importance of precise communication, contextual reasoning, and analytical capabilities in interpreting and responding to gesture-based interactions.

3.3 Context Management Agent

The Context Management Agent plays a crucial role in our system, offering intelligent management of various types of context information. Unlike static, rule-based context management system, this agent dynamically adapts to unfamiliar context thanks to LLM’s vast and intricate knowledge base, ensuring a flexible and responsive interaction environment.

A dedicated Context Management Agent represents a significant step forward in the evolution of gesture-based interfaces. By segregating context handling from the inference processes, we streamline system operations and enhance the overall system’s ability to effectively utilize diverse contextual data. This separation allows for more sophisticated and nuanced interaction paradigms, where gestures can be interpreted in varying scenarios with greater precision. Moreover, this

architecture facilitates easier updates and scalability, as new contexts or rules can be integrated without disrupting the core inference mechanisms. Consequently, this approach not only addresses previous limitations but also sets a new standard for the development of adaptive, efficient, and user-centric gesture recognition systems. The overall prompt is structured into two parts shown in Figure 7.

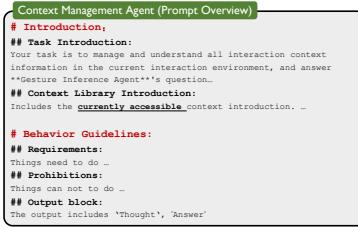


Fig. 7. Prompt for Context Management Agent.

The introduction includes a Task Introduction and a Context Library Introduction. The latter introduces the currently accessible contexts and is designed to be adjustable. Similar to the Gesture Inference Agent, iterative implementation guides the Context Management Agent's requirements and prohibitions, with 4 requirements and 2 prohibitions. The output is structured in JSON format for easy parsing, allowing efficient information extraction and facilitating subsequent conversation rounds.

3.3.1 Context Library Operations The Context Management Agent supports three main operations within the context library: **Adding** new context types, **retrieving** context information based on queries and **calculating** specific context values.

- **Adding Context Types:** Context types and their values are organized using markdown for text and JSON for structured data. New context types are introduced with natural language descriptions, and associated values are formatted in JSON. This facilitates easy system expansion and automatic incorporation of new contexts into the operational prompt.
- **Retrieving Context Information:** The agent automates retrieval operations, streamlining the process by organizing data in a standardized format (e.g., JSON), which the LLM can autonomously interpret.
- **Calculating Context Value:** The agent can calculate specific context values based on predefined methods in the context's description. These methods can be implemented in a Python script. Upon activation of a calculation operation, the agent generates a unique placeholder. This placeholder triggers the system to execute the Python script, automating the retrieval of context values and producing the desired output. The placeholder is then replaced with this output in the final response.

The design of Context Management Agent offers significant benefits, such as the ease of adding new context types using simple natural language descriptions and the automated retrieval and understanding of context information. This enhances the system's versatility and usability. Our design also inform the design of context management components of future interactive systems.

4 Evaluation

We designed two experiments to assess GestureGPT's adaptability and effectiveness under different interaction scenarios with varying context environments and camera perspectives.

In the first interaction scenario, a user controls smart home appliances through an AR headset. In this setup, gestures are captured from the user's own viewpoint, offering a first-person perspective.

It is a key interaction scenario in the future when users interact with environment using head-mounted devices.

Our second interaction scenario mimics the case when a user is watching online videos. The user watches the video on a monitor with a camera capturing gestures from the third-person perspective. This setup is prevalent in a variety of settings, including smart TVs, interactive public displays, educational environments, gaming, *etc.* The complexity increases in this scenario, as there are plenty of functions and interactive elements on a webpage.

We selected four contexts in our experiments to evaluate GestureGPT’s performance in different context settings:

- **Interface Function List:** Crucial for mapping gestures to interface functions, this context includes interface name and a list of functions with their names, locations, and unique IDs, key for navigating the user’s interaction environment.
- **Gaze Information:** Data on the user’s gaze, given in 3D (in home scenario) or 2D (in video scenario) coordinates.
- **Interaction History:** A record of the user’s recent interactions.
- **External Context Information:** Information from other devices or sensors. We introduce this type of context to explore how other factors impact gesture understanding and whether the agent can leverage this information.

We informed participants that both video and eye movement data would be collected during the study, and we assured them of the confidentiality and safety of their data. The study is IRB approved by our local institution. All participants provided informed consent. In both experiments, we asked participants to perform the gesture as they see fit to finish the task. Both static hand poses and dynamic gestures involving hand movements were permitted.

We employed the OpenAI GPT-4 model as the underlying architecture for our triple-agent system. Specifically, we utilized the gpt-4-1106-preview version for our evaluations. We configured the request temperature to 0 so that the model’s output is as consistent as possible. Apart from this, we adhered to OpenAI’s default settings for other parameters. To mitigate the effects of randomness and enhance the reliability of our findings, we ran the experiment repeatedly for three times. The aggregated results from these iterations were used to substantiate our conclusions.

4.1 Experiment 1: Augmented Reality-Based Smart Home IoT Control

This study explores the interaction between users and smart home devices through augmented reality (AR), specifically using gesture controls in a simulated kitchen environment. Participants perform gestures as they see fit to control various IoT devices, triggering changes in device state accordingly.

4.1.1 Experiment Setting and Procedure

- **Experimental Platform and Data Collection** - The Microsoft HoloLens 2 served as the primary experimental platform, offering APIs to capture user gaze and hand gesture data accurately. The experimental environment was developed using Unity (version 2020.3.24f1), the Mixed Reality Toolkit (MRTK 2.8.0), and the OpenXR Plugin (1.7.0). The devices were represented by 3D models anchored in the space: a light, a smart cabinet, a smart screen, an oven, and an air cleaner.
- **Context Library Setup** - The experiment implemented context library as follows:

- *Interface Function List*: Drawing from the XiaoMi SmartHome API¹, five devices and their corresponding functions were synthesized to form a function list. Each device has 3-5 functions with a total of 18 functions in this scenario.
- *Gaze Information*: User gaze data was captured using the HoloLens 2 Gaze API and saved as 3D spatial coordinates.
- *Interaction History*: Interaction history was extracted from the task sequence.
- *External Context Information*: There might be context information that is external to our system, which can significantly impact the grounding reasoning. We defined several external contexts corresponding to different tasks to understand if our system can correctly leverage those.
- **Task Descriptions** - Eight tasks were designed to simulate smart device control (Table 6).
- **Participants** - We recruited 16 participants from three local schools, compensating them at a rate of \$12 per hour. Their ages ranged from 15 to 35 years (MEAN = 26.625, SD = 5.325), comprising 13 males and 3 females. Participants included 1 high school student, 4 undergraduate students, 9 graduate students, and 2 research engineers. None of the participants had prior experience with AR/VR devices, which helped minimize bias due to varying familiarity with such technology.
- **Task Procedure** - Upon their arrival, participants were briefed about the scenario and the devices involved. They are asked to make any gesture deemed most intuitive using the right hand after raising their hand above their chest to initiate the trigger. A preliminary warm-up session was conducted to familiarize the participants with the AR devices and gesture control operations. Following this, they were instructed to complete the eight tasks. Feedback from the devices was simulated to enhance the interaction experience and realism of the study.

Detailed information on the experiment simulation interface, the list of devices and their functions, and the task list are provided in Appendix B.1. A total of 16 participants × 8 tasks = 128 gestures were collected.

Table 2. Main Results of GestureGPT in the Two Experiments

	Smart Home Scenario				Video Streaming Scenario			
	Top 1 (↑)	Top 3 (↑)	Top 5 (↑)	Negative (↓)	Top 1 (↑)	Top 3 (↑)	Top 5 (↑)	Negative (↓)
Random Guess	5.56%	16.67%	27.78%	72.22%	3.15%	9.46%	15.76%	84.24%
Baseline	10.94% \pm 3.38	24.48% \pm 3.51	35.16% \pm 2.92	64.84% \pm 2.92	19.53% \pm 3.55	38.28% \pm 1.10	54.43% \pm 1.84	45.57% \pm 1.84
Only Gaze	35.16% \pm 1.28	70.05% \pm 1.33	83.59% \pm 1.10	16.41% \pm 1.10	25.78% \pm 0.64	47.66% \pm 3.19	60.42% \pm 2.88	39.58% \pm 2.88
Only History and External	23.18% \pm 4.25	37.50% \pm 4.47	49.48% \pm 4.34	50.52% \pm 4.34	26.30% \pm 3.01	47.14% \pm 5.12	63.28% \pm 3.38	36.72% \pm 3.38
All	44.79% \pm 3.21	67.45% \pm 4.10	79.69% \pm 1.69	20.31% \pm 1.69	37.50% \pm 4.18	59.90% \pm 4.83	73.44% \pm 1.91	26.56% \pm 1.91

4.1.2 Results Analysis We evaluate GestureGPT’s performance by running the collected data through our system under four different context settings: 1) the baseline with only the function list (*Baseline*), 2) with gaze information (*Only Gaze*), 3) with interaction history and external information (*Only History and External*), and 4) all contexts are available (*All*). For comparison, we also provide results from *Random Guess*, which randomly selects one from all functions. We repeat the tests three times under each setting to ensure robust conclusions. The main results are presented in the left part of Table 2, and a corresponding illustration is provided in Figure 8.

Our first observation is that GestureGPT can effectively utilize the context information to determine the exact intention of the users. The accuracy is effectively improved to an impressive

¹<https://iot.mi.com/new/doc/design/spec/xiaomi>

extent when either gaze or history and external is incorporated. When all contexts are combined together, the overall framework achieves the best Top-1 performance at 44.79%.

When comparing the benefits of gaze versus history and external information, we empirically find that gaze significantly outperforms history and external information. As shown in Table 2, the *Only Gaze* setting achieves 35.16%/83.59% at Top-1/Top-5 accuracy, outperforming the 23.18%/49.48% achieved by the *Only History and External* setting by a considerable margin. We attribute this to the semantic similarity between candidate functions shared by different home devices and their broad spatial distribution across the entire room space. These two factors significantly increase the reasoning difficulty under the Smart Home scenario. Facing such challenging situations, GestureGPT can fully exploit the gaze information as well as its own commonsense knowledge to finally locate and zoom in on possible candidates through multi-step, collaborative spatial reasoning. As explained in Section 3.3.1, when the Gesture Inference Agent requests gaze information, the Context Management Agent not only outputs the gaze coordinates but also identifies the relevant device within the gaze path using external tool-augmented Python scripts.

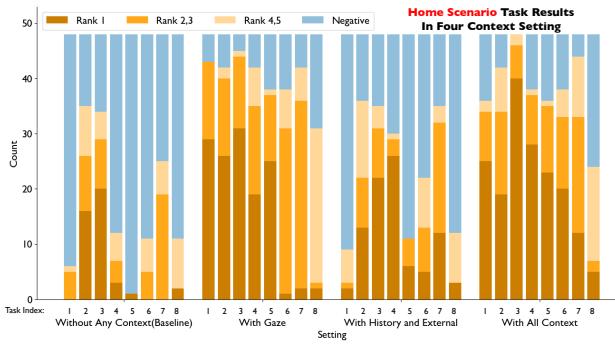


Fig. 8. Illustration of results from Experiment 1: Home Scenario, encompassing all 4 settings (across the x-axis) and 8 tasks (indicated by hue). For each setting, the 8 tasks are arranged from left to right as follows: *task 1 Unlock Cabinet*, *task 2 Increase Light Brightness*, *task 3 Next Recipes*, *task 4 Open Oven*, *task 5 Open Air cleaner*, *task 6 Set Timer*, *task 7 Switch Input*, and *task 8 Phone Call*. Each specific bar is further divided into Top 1 / Top 3 / Top 5 accuracy (**Top 1**, **Top 3**, **Top 5** in gradient brown) and Negative outcomes (in blue), totaling 48 data points.

Moreover, while integrating all context achieves the highest Top-1 performance, it unexpectedly underperforms in the Top-5 metric compared to the Gaze Only setting (79.69% vs 83.59%). Through further case analysis, we hypothesize that an excess of context can sometimes disrupt the agent’s analysis, leading to the following behaviors: 1) Irrelevant context information can at times mislead the agent. For example, the mention of “fingerprint unlocking” (in task 1) within external information might lead the agent to infer fingerprint recognition as the unlocking method, rather than a gesture. 2) An abundance of context causing the agent to overlook certain information. For example, in the “Open the air cleaner” task (in task 5) where the agent wrongly assumes the device is on, based on external information “The air purifier’s sensor detected heavy cooking fumes,” while ignoring its actual OFF status. These insights will be further discussed in Section 5.4 regarding optimal context selection.

Finally, we observe that Tasks 2, 3, and 7 exhibit relatively high baseline performance, all involving operations related to “switching”. Finally, we observe that Tasks 2, 3, and 7 exhibit relatively high baseline performance, all involving operations related to “switching”. Similarly, for Tasks 3 and 7, GestureGPT employed a comparable analytical approach to discern the correct answer. This

demonstrates GestureGPT's strength in analyzing gestures alongside current device states, thereby accurately interpreting user intentions even in the baseline context.

In conclusion, our investigation into GestureGPT's performance across different contexts underscores the significant enhancements brought about by gaze data and the contributions of history and external information. Further refinement in how external information is presented and queried will unlock greater performance of our system.

4.2 Experiment 2: Online Video Streaming on PC

This scenario is set when a user is watching online video on a PC monitor. Grounding gestures to the correct function is much more challenging in this scenario compared to the previous one. The video streaming interface contains a considerably broader range of functions, with numbers up to 66 in some tasks from the previous 18 functions. Moreover, many functions have similar semantic meanings, such as the “vlog channel” button and the “anime channel” button, making them difficult to distinguish solely through gestures. Furthermore, the interactive buttons and elements on the screen are much smaller than the smart appliances in the previous experiment, which reduces the performance of gaze-based function differentiation. The size of function elements ranges from 0.27 to 20.71 cm² (MEAN = 2.73, SD = 3.67). By navigating through an interface rich in functions and semantic complexities, we intend to explore the boundary of our system’s performance and understand whether it can differentiate user intentions with only minor differences, which is essential for real-world applications.

4.2.1 Experiment Setting and Procedure

- **Experimental Platform and Gesture Data Collection** - Our experimental framework utilizes Python and Selenium² to interact with a video streaming platform, specifically targeting the website “[China] From the Spring and Autumn Period to the Prosperous Tang Dynasty (Season 1, 12 Episodes)” on Bilibili³. The platform automates video control operations via Selenium. User gestures are captured using a 1080P resolution webcam.
- **Context Library Configuration** - The study incorporates a comprehensive context library comprising four distinct aspects same as in previous scenario:
 - *Interface Function List*: The function list is automatically extracted from the webpage and organized. Functions on the website included their position and raw HTML code are identified via JavaScript. The code for each function is then extracted and fed into GPT-4 to generate function names, aiding in the compilation of the interface function list. For tasks 4, 5, and 6, the function count is 17; for all other tasks, it is 66.
 - *Gaze Information*: The Tobii Eye Tracker 5 is used to collect gaze data from participants.
 - *Interaction History and External Context Information*: Interaction history was extracted from the task sequence, while the external contexts were predefined.
- **Task Descriptions** - Eight tasks were designed to simulate common operations performed while watching videos (Table 8).
- **Participants** - We recruited 16 participants from four local schools, compensating them at a rate of \$12 per hour. Their ages ranged from 18 to 35 years (MEAN = 26.875, SD = 4.689), comprising 10 males and 6 females. Participants included 5 undergraduate students, 8 graduate students, and 3 research engineers. None of the participants had prior experience with AR/VR devices, which helped minimize bias due to varying familiarity with such technology.
- **Task Procedure** - Participants were briefed on the experiment’s aims and the devices utilized upon their arrival. They were also asked to make any gesture deemed most intuitive with their

²<https://www.selenium.dev/>

³<https://www.bilibili.com/video/BV1sh41j7A4/>

right hand, similarly triggered by raising their hand above the chest. There was a warm-up phase for the participant to familiarize with gesture controls. Following this, the participant sequentially completes the eight tasks. Specific gestures triggered predefined responses on the website, simulating real-time interaction for a more realistic experience.

As a result, a total of $16 \text{ participants} \times 8 \text{ tasks} = 128$ data points were collected. Detailed information about the experiment simulation interface, the list of devices and their functions, and the task list is provided in Appendix B.2.

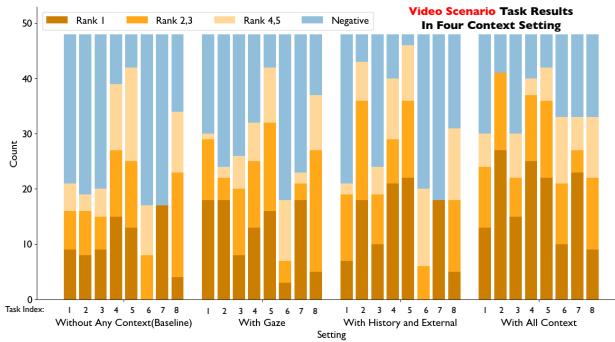


Fig. 9. Illustration of results from Experiment 2: Video Scenario, encompassing all 4 settings (across the x-axis) and 8 tasks (indicated by hue). For each setting, the 8 tasks are arranged from left to right as follows: task 1 *Adjust Column*, task 2 *Video Progress Control*, task 3 *Enter Fullscreen*, task 4 *Pause Video*, task 5 *Play Video*, task 6 *Exit Fullscreen*, task 7 *Like the Video*, and task 8 *Next Video*. Each specific bar is further divided into Top 1 / Top 3 / Top 5 accuracy (Top 1, Top 3, Top 5 in gradient brown) and Negative outcomes (in blue), totaling 48 data points.

4.2.2 Results Analysis

In this scenario, we evaluate GestureGPT under the same four context settings. The results are presented in the right part of Table 2 as well as Figure 9.

The main conclusion that GestureGPT can effectively incorporate various context information to reason and predict user intentions remains consistent. Specifically, Top-1 accuracy improved from 19.53% to 37.50% when all types of context are available, significantly outperforming each single context setting: +11.20% compared to only the history and external setting, and +11.72% compared to the only gaze setting. Both contexts contribute to the final performance, and it greatly drops if either is excluded. Compared with Experiment 1, where gaze information brings relatively more utility, these results further demonstrate that it requires joint reasoning over all available contexts to achieve the best performance under such a more complicated scenario.

An intriguing observation from our results was that the baseline performance was superior to that observed in the home scenario, thereby necessitating task-specific exploration. Tasks such as pausing, playing videos, and selecting the next video (Tasks 4, 5 and 8), which require less context, exhibited strong performance across all tested context settings. This phenomenon suggests that the agent's inherent knowledge of video streaming interfaces makes it likely to infer actions commonly seen in such a video streaming interface. To some extent, the agent's success in identifying these operations can be attributed more to an educated guess rather than a calculated match of gesture and context, indicating a form of intuition derived from the model's extensive training data. This intuition helps our system's performance in certain tasks, but also acts as bias in other tasks. This phenomenon is discussed further in Section 5.5.

Another special case is Task 7, which is semantic specific task, highlights the essential role of accurate gesture description in achieving precise gesture grounding. Out of 16 participants, 12 performed the “thumbs-up” gesture for the task. In such cases, if the agent accurately recognizes the gesture, the outcome is correct, notably reflecting in high Top-1 accuracy results. The failures predominantly occurred due to incorrect gesture segmentation. For example, because users formed a fist after lifting their hand, leading the agent to mistakenly focus on the action of making a fist, which leads to map the “Full Screen” or “volume” function. We conducted a pilot validation exercise on the gesture segmentation rule used in our system with a human labeler splitting gesture frames from the whole video on data from eight participants as the ground truth in gesture segmentation. Our results, compared to the ground truth, revealed a high recall rate of 95.28%, yet a precision of only 43.91%, reflecting the inclusion of non-gesture-related frames. Future enhancements could involve advanced object detection techniques , differentiating interactive vs non-interactive gestures, or a specially designed beginning gesture to improve segmentation precision.

In cases where gesture descriptions were poorly articulated, incorporating additional context proved beneficial. For example, considering interaction history context, such as users exiting full-screen mode, allowed the agent to infer that the user might want to perform operations unrelated to video control, like “liking” a video. This various context adaptation significantly improved this task performance in Top-5 accuracy metrics in this task from 35.42% to 68.75%.

Despite the challenges we designed in the video streaming scenario, GestureGPT still demonstrates commendable performance, largely attributed to the LLM’s robust common sense and contextual interpretation capabilities.

4.3 Result Analysis: Gesture Description Quality Assessment

The quality of the Gesture Description Agent was evaluated through an expert questionnaire. For this evaluation, we randomly sampled descriptions generated from three repetitions for 256 gestures across two scenarios, resulting in a compilation of 256 gestures and their descriptions. The same questionnaire was then distributed to two gesture experts, who rated each gesture description on a 5-point Likert scale (1 being “Almost no description is correct or relevant to the gesture” and 5 being “Almost all of the description is correct or relevant to the gesture”). Both expert have published gesture-related research articles on premier conferences. One expert scored 3.28 (std = 1.41), and the other scored 3.74 (std = 0.73). Their Kendall’s W score was 0.853, indicating a high level of agreement between the two experts. The positive ratings show that our description agent can capture key information of the gesture.

4.4 Pilot Study: Human Performance in Gesture Understanding

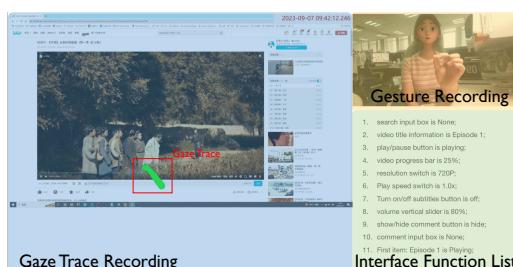


Fig. 10. Pilot study interface showcasing. On the left, a video stream with a highlighted gaze trace indicating user focus during the task. On the right, a gesture recording from a video streaming scenario alongside a detailed interface function list. This setup was used to assess participant ability to map gestures to corresponding interface functions, facilitated by context such as gaze trajectories and historical information.

To gauge human performance on tasks similar to those managed by GestureGPT, we conducted a pilot study with four participants who were unfamiliar with the system and its experimental setups. We randomly selected data from four users involved in Smart Home and Video Streaming scenarios. During the pilot, participants were provided with video recordings that included the user’s gesture poses and movements, along with a list of interface functions available at that time. They were tasked with mapping each gesture to its corresponding function. The pilot study interface as shown in 10.

Initial attempts by participants were guided by *gaze context*, as demonstrated through gaze trajectory recordings, and *historical context*. Subsequent attempts were further supported by providing *external context*, the pilot study interface shown in 10. Subsequent attempts were further supported by providing *external context*. Results revealed significant challenges in gesture understanding even for humans: initial Top-1 and Top-5 accuracies were 43.8% and 81.3% for the smart home, and 18.8% and 50% for video streaming, respectively. With added external context, accuracies increased notably to 58.3% (Top-1) and 91.7% (Top-5) in the smart home, and to 83.3% (Top-1) and 100% (Top-5) in video streaming. This preliminary result underscores the inherent complexity of interpreting gestures, even for humans, in contextually rich settings. So compared with gesture recognition, gesture understanding is much more difficult and challenging.

5 Discussion

5.1 Integration and Utilization of Top-N Predictions

Due to the inherent complexity of gesture understanding—far surpassing that of mere gesture recognition—we approach this challenge as a recommendation task, which is why we show the Top-3 and Top-5 accuracies. GestureGPT currently outputs a Top-5 candidate function list, which can be readily augmented with list selection interaction optimizations, as studied in many existing works. Specifically, Isomoto et al. [25] developed a dwell selection system that employs machine learning to predict user intent solely based on eye movement data. Furthermore, statistical design of dynamic menus could also expedite user selection particularly in AR/VR contexts. G-Menu [63] utilizes gestures and keywords to design a dynamic menu, aiding in swift function selection. These techniques and designs can be readily integrated into GestureGPT to achieve efficient user selection of the target function. The rapid advancement of LLMs and their increasing capability as contextualized agents, LLM-driven frameworks like GestureGPT will provide improved performance so that its Top-1 accuracy reaches a directly applicable level, ultimately ease the burden of secondary selection of users.

5.2 Language Input (LLM) vs. Visual Input (LMM) for Agents

GestureGPT is currently built with a Large *Language Model* for its superior understanding and reasoning capability. For the Gesture Description Agent, an alternative implementation could employ a Large Multimodal Model (LMM), which seems more intuitive. So we tested replacing the rule-based Gesture Description Agent with GPT-4o’s vision in a video streaming scenario. With *All Context* considered, the Top-1 and Top-5 accuracy results were 41% and 72%. These results demonstrate performance comparable to our rule-based module which is 37.5% and 73%, thereby validating the effectiveness of the rule-based approach.

On the other hand, LMMs require users to upload gesture recordings, which raises further privacy concerns. By contrast, our solution can process the gesture input with end-affordable devices and only send anonymized skeleton signals to the data center, where LLMs can perform dense computing. Nevertheless, it is anticipated that with further development of LMMs, GestureGPT could integrate

both the visual recognition capabilities of LMMs and the prominent commonsense understanding, contextual reasoning capabilities of LLMs, to ultimately better serve its purpose.

5.3 Single Agent vs. Multi Agent

Assuming privacy concerns regarding LMMs are not considered, employing gesture visuals with contextual information in a LMM-based single-agent architecture could significantly reduce the system delay. Unlike multi-agent systems, which require multiple reasoning cycles to interpret gestures, a single-agent approach can deliver immediate results within one iteration. We conducted a preliminary test with data from one user in the smart home scenario, and the LMM-based single-agent approach achieved a Top-1 accuracy of 25.0% and a Top-5 accuracy of 75%. Due to privacy concerns, comprehensive testing was limited because other users did not authorize the publication of data.

Although the end-to-end LMM approach exhibits shorter delays, it presents several disadvantages: it is cumbersome to adjust and train, lacks task transferability, and falls short in terms of scalability and generality. Conversely, the multi-agent architecture offers significant advantages, such as enabling component reuse, and facilitating modular training and optimization. Each architecture has its own applicable scenarios, and together, they can complement each other to achieve superior results.

However, the multi-agent architecture of GestureGPT inherently offers greater flexibility and is designed for broad applications. For example, instead of cameras, the description agent can work with sensors like IMUs, electromyography, and LiDAR, provided they can reconstruct the hand pose to some extent. The framework handles complex interaction contexts to mimic human gesture understanding, such as interpreting gestures linked to ongoing events. Integrating context into the inference agent would greatly increase the workload and limit flexibility. The independently implemented modules allow for easier optimization. Smaller models within the architecture are plug-and-play, thereby avoiding constant end-to-end training.

Furthermore, we contend that the multi-agent architecture of GestureGPT is ideally suited for complex interaction scenarios where single-step reasoning is insufficient. For example, an agent might need to interact with users or external databases before making decisions. Our architecture supports distributed processing, enabling different agents to handle specific tasks asynchronously, thereby having the potential to further enhance efficiency in managing complex interactions. For a detailed discussion on how this architecture addresses system delays, see Section 6.2.

5.4 Context Selection in Complex Systems

Our experiments revealed that although adding more contexts is supposed to be informative and useful, for certain circumstances, it might also bring overwhelming irrelevant information and accordingly distract the reasoning process of LMM agents (See Top-5 accuracy of *All* setting vs *Only Gaze* setting in Smart Home scenario). In Smart Home scenario, most informative contexts are concentrated on gesture itself and gaze, while history and external provide relatively less contribution to the final performance.

On one hand, the inclusion of more heterogeneous contexts increases the complexity and difficulty of context organization and management, which then requires more competent agents to process and reason over them. On the other hand, contexts with less information entropy inevitably brings irrelevant implications, and may result in misleading of agents for specific cases. This underscores the importance of making informed trade-offs between context incorporation and agent capability to optimize the system performance.

5.5 LLM Common Sense Bias

This phenomenon pertains to the cognitive biases of LLM, which has been recognized as a common issue that influences LLM outputs [53].

Such a bias was also observed in our system, notably within the video streaming scenario. When the agent learned that this is a video streaming scenario, there is a bias towards predicting video control functions that are commonly used on such an interface, such as play, pause, and volume control. On one hand, even in the absence of context, the agent can make accurate guesses if the intended function is one of such functions. On the other hand, it leads to the consistent inclusion of these functions within the candidate options, detrimentally impacting the top-1 accuracy.

One way to address this bias could involve employing Modular Debiasing Networks [13] to mitigate bias or utilizing sophisticated prompt engineering to diminish its effects, which we plan to investigate in the future.

5.6 System Scalability

By substituting the Gesture Description Agent with specially designed counterparts, our system can adapt to more input modalities and more generalized form of gestures.

In our implementation, gesture feature extraction is solely based on a RGB camera and MediaPipe. Yet this approach is susceptible to lighting conditions and finger occlusion issues. Wearable devices provide an alternative robust solution for hand reconstruction [22, 73, 78], which even works with subtle gestures like thumb-tip gestures [19] and wrist gestures [17, 18]. Hand landmarks reconstructed from wearable sensors can then be used to generate gesture description.

Our system can also be extended to general gestures, beyond the scope of merely hand gestures. For example, touch-screen gestures can be extracted as traces and pressure intensity, which differ significantly from the form of hand gestures. But a specially designed Gesture Description Agent can extract the features of such gestures (either using rule-based methods or leveraging large language or vision models) and describe it to Gesture Inference Agent, thus easily integrated into our framework.

6 Limitation and Future Work

6.1 Handling Left-Handed Gestures, Multiple Hands, and Non-Interactive Gestures in Frame

GestureGPT currently identifies meaningful gestures when a right hand is raised above chest level. However, this implementation assumes that only one hand, typically the right hand, is visible. This assumption is often unrealistic, as interactions in real environments may involve left-handed users or require two-handed gestures, thereby limiting the system's inclusivity. Furthermore, this simplification can lead to false activations, as unintended gestures or non-interactive movements may inadvertently trigger the system. This issue is exacerbated in scenarios where various non-interactive hand movements and multiple hands are present within the camera frame, potentially confusing the model and degrading system performance. Evaluating the system's robustness when both hands are visible or when left-handed gestures are involved is essential for fully understanding its practical capabilities. Future iterations of our system will focus on distinguishing between intentional gestures and extraneous hand movements, even in the presence of multiple hands, and on improving left-hand gesture recognition to better accommodate left-handed users.

Several existing studies have addressed these challenges. Research [29] has explored techniques for distinguishing between the static and dynamic aspects of hand movements, along with the challenges of recognizing one- and two-handed gestures. Similarly, vision-based interaction approaches for augmented reality [60] emphasize the importance of simplifying hand models and applying user

intention recognition methods to achieve robust performance in complex scenarios. Additionally, frameworks that incorporate body pose, gaze direction, and multimodal feedback mechanisms have shown promise in predicting interaction intent [12, 54], while other studies have integrated EEG and gaze patterns to differentiate between meaningful and irrelevant gestures during tasks [16, 33].

The architecture of GestureGPT is designed to be adaptable and scalable, enabling the integration of these advancements to further enhance its capabilities. Incorporating a left-hand recognition module within the Gesture Description Agent would allow the system to support both left-hand and two-handed gestures. Additionally, expanding the contextual input processed by the Context Management Agent could improve the system's ability to distinguish non-interactive gestures, thus reducing the likelihood of false activations. In the future, we plan to explore these challenges further, including validating the accuracy of left-hand gesture recognition, providing comprehensive support for two-handed gestures, and refining the system's activation mechanisms to enhance its practicality and robustness.

6.2 System Delay and Applicability

The overall time cost of the framework is primarily driven by LLM-related processes, accounting for up to 90% of the total time expenditure. These costs include LLM API request costs. For each gesture understanding task, GestureGPT typically requires 1 request for the gesture description and 6 rounds of conversation between the Gesture Inference Agent and Context Management Agent, highlighting the significant impact of response delay on the real-time performance of our system. In its current conceptual form, without a focus on practicality, our framework averages 227 seconds and 38,785 tokens per task.

Although the conceptual framework of GestureGPT demonstrates promising potential for HCI applications, the current implementation faces significant challenges related to system delays, which affect its real-world applicability. Addressing these delays is essential for transitioning GestureGPT from a theoretical model to a practical tool capable of functioning in dynamic, real-time environments. Possible directions for improvements include:

- **KV-cache:** API calls require concatenating long system prompts in each round, exponentially increasing time costs. Our current implementation appends the entire context library (11,000 tokens) to each query. Implementing a KV-cache would reduce the need to reprocess identical contexts, potentially decreasing interaction time to approximately 20 seconds per task. This enhancement is expected to streamline operations by preserving computed attention values for recurring queries.
- **High-Performance Hardware Utilization:** Integrating advanced computational resources, such as Groq's hardware, could further reduce processing times. Groq's fast model inference services offer an output speed of over 700 tokens per second.

By combining Groq and KV-cache, the first context round would take 2-3 seconds, with subsequent rounds taking a total of about 1-2 seconds (averaging 6 dialogue rounds, each with 200 tokens input and 100 tokens output). This would reduce the theoretical optimal latency to under 5 seconds. Although we do not currently have access to KV-cache-enabled high-performance LLMs or the computing acceleration hardware provided by Groq, we anticipate that GestureGPT will attract interest from other research domains, fostering collaborative efforts towards developing a practical and accessible free-form gesture interface.

On the other hand, since each agent of our system has specialized and clear goals, we believe a properly fine-tuned LLM could mirror GPT-4's effectiveness with fewer resources as demonstrated in NLP tasks [84]. Besides, advances in model distillation and acceleration hardware suggest that

lightweight LLMs could be deployed directly on devices with fast inference [38, 82, 82] for real-time performance in the future. Under which circumstances, the API request cost can also be removed.

While the current system delays may limit immediate real-world applications, the outlined advancements and potential collaborations offer a promising path toward minimizing these delays. Future iterations of our system, supported by ongoing research and technological improvements, could effectively reduce latency to levels suitable for dynamic, real-time user interactions. Such developments would not only address the practical limitations discussed but also pave the way for broader adoption and impact across various domains, ultimately fulfilling the potential of real-time, intuitive gesture-based interfaces.

7 Conclusion

In this work, we introduce GestureGPT, a human-mimicking framework for understanding free-form hand gestures, which capitalizes on the capabilities of LLMs. Unlike traditional gesture recognition systems that require predefined gestures, GestureGPT automatically maps spontaneously performed, free-form gestures to their targeted interface functions without any user training. This is achieved through a collaborative framework involving three specialized agents: a Gesture Description Agent, a Context Management Agent, and a Gesture Inference Agent. These agents work together to interpret gestures, manage context, and apply common sense reasoning to accurately discern user intents.

Our offline evaluations in two realistic scenarios demonstrated promising results, achieving the highest zero-shot Top-5 gesture grounding accuracy of 83.59% for smart home control and 73.44% for video streaming. These outcomes highlight the significant potential of GestureGPT as a conceptual model for future gesture-based interaction systems. They underscore the viability of our approach in reducing the cognitive load on users by eliminating the necessity to learn, memorize, or manually link gestures to functions.

As a conceptual framework, GestureGPT sets the groundwork for future advancements in natural interaction interfaces. While not yet a practical system ready for everyday use, our framework serves as a foundational step towards realizing more adaptive and intuitive ways to interact with technology. Future research could focus on refining this paradigm, enhancing its responsiveness, and expanding its applicability to include a broader array of natural user interfaces supporting different modalities like facial expressions, body movements, and postures, unleashing the full potential of the framework proposed by GestureGPT.

Acknowledgments

The authors would like to thank Benfeng Xu, whose insightful comments and suggestions were invaluable in writing this paper. This work was supported by the Hunan Provincial Natural Science Foundation of China (No. 2023JJ70009), the HNXJ Philanthropy Foundation (KY24017), the Science and Technology Innovation Program of Hunan Province (No. 2022RC4006), and the Young Scientists Fund of the National Natural Science Foundation of China under Grant No. 62102401.

A Gesture Description Rules

A.1 Rules Calculation Method

Flexion of a finger is calculated as the total bending angle of each joint. For thumb it is the bending angle of the ip joint, and for other fingers, it is the bending angle of the pip and dip joint. Then, two parameters *straight_threshold* and *bent_threshold* are set to determine if the finger is straight, bent, or “unsure” if the result falls between them. Since thumb has a different joint structure compared with other fingers, a new pair of thresholds are specially set for thumb.

Proximity of two fingers is calculated as the average minimal distance from each finger’s joint to the other finger. Two thresholds *i.e.*, *together_threshold* and *separated_threshold* are set to determine if the two fingers are pressed together, separated, or “unsure” if the result falls between them.

Contact of thumb and another finger is computed as the distance between their fingertips. Then, two thresholds, *i.e.*, *contact_threshold* and *not_contact_threshold* are set to determine if the two fingers’ fingertips are in contact or not, or “unsure” if the result falls between them.

Point direction of thumb is computed as the direction from thumb’s mcp joint to tip joint. Then, it is compared with two reference vectors representing upward and downward. If a reference vector has the minimal angle with the palm orientation vector and the angle is below *angle_threshold*, the reference vector would be the thumb’s pointing direction. Note that it is only applicable when thumb is straight. If thumb is bent, the result is set to “unsure”. This is especially useful when discriminating between gestures like “thumb up” and “thumb down”.

Palm orientation is computed as the direction to which the palm is facing. It is computed by the cross product of two vectors within the plane of the palm (Figure 4). Then, the direction is compared with six reference vectors representing upward, downward, left, right, inward and outward. If a reference vector has the minimal angle with the palm orientation vector and the angle is below *angle_threshold*, the reference vector would be the palm orientation. Otherwise the orientation of palm is set as “unsure”.

Hand position is computed as the geometrical center of a hand by taking average of all 21 landmarks’ coordinates. No parameter or threshold is applied here.

A.2 Training for Threshold and Evaluation for Gesture Description Rules

We use HaGRID [30] dataset to tune and evaluate our rules. HaGRID contains images of 18 kinds of gestures (call, dislike, fist, four, like, mute, ok, one, palm, peace, peace inv., rock, stop, stop inv., three, three 2, two up, two up inv.).

A.2.1 Training Process A subsample split of HaGRID has 100 images per gesture, and it is used to tune the rule parameters.

To tune the rule parameters, we first manually annotate the ground truth label for each rule on each gesture class. For each gesture, we assume that most people perform it in the same way, and thus the ground truth label is obtained by common knowledge. (For example, for “thumb up” gesture, we label thumb as straight, index to pinky fingers as bent, index and middle finger are pressed together, *etc.*)

However, there exist ambiguous cases, in which more than one labels are acceptable. (For example, in “peace” gesture, it is reasonable for thumb to be either straight or bent.) In this case, the label is more than one, and consequently cases like this are not used for parameter tuning.

Most rules are tuned using the whole subsample split. One exception is **Finger Flexion** on thumb, because in most gesture classes, the ground truth of thumb is either [“straight”] or [“straight”, “bent”] (thus not used for training), and only one class is [“bent”]. To address the issue of severe imbalance for thumb flexion, we specially select the training set, which is composed of gesture “like” (thumb is straight) and gesture “ok” (thumb is bent), and each image in this set is manually checked to ensure the quality of the training data.

As is shown above, for those cases used to tune the rules, the ground truth label is one of the two possible states (*e.g.*, “straight” or “bent”). But during prediction, we use Three-Way Decision Making method [76]. When the rule cannot clearly determine the state, it will predict it as “unsure”, to avoid wrong conclusions that may easily mislead LLMs. Consequently, different from the classical

binary classification, some modifications have been made to the correctness assessment and tuning criteria to accommodate our specific setting.

Correctness Assessment. For each (prediction, label) pair, there are three possible circumstances:

- Unsure: Prediction = “unsure”.
- Error: Prediction \neq “unsure”, and prediction \neq label.
- Correct: Prediction \neq “unsure”, and prediction = label.

Tuning criteria. To find the optimal parameters, we define the loss in different cases as follows:

- Unsure: Loss = 0.2
- Error: Loss = 1
- Correct: Loss = 0

In “unsure” cases, the loss is between 0 and 1. This is because if we set the loss to be 0 (same as “correct” case), then the rules tend to predict every case to “unsure”; if we set the loss to be 1 (same as “error” case), then the rules tend to not predict any “unsure”, which may lead to misleading predictions.

A grid search is conducted to find out the optimal parameter with minimal average loss. The optimal parameters for each rule is shown in Table 3.

A.2.2 Test Performance We test the generalization ability of our rules on two dataset: HaGRID test set (third-person view) and EgoGesture dataset (first-person view) [80].

The performance on HaGRID test set (38576 images) is shown in Table 3. For most rules, the average error rate among all gestures is below 4.2% and the overall accuracy is above 88.2%. One exception is flexion rule for thumb, in which the output “unsure” takes about 45% of all cases. This may be attributed to the unique shape of thumb: the topmost segment of the thumb is curved by nature, so when the thumb is extended, the landmark seems to be slightly bent, which may affect the train and test process. But by a three-way decision, we use “unsure” to avoid the misleading information that may potentially confuse the Gesture Description Agent.

When checking the algorithm’s performance for each gesture, it is shown that the error rates for most gestures are below 5% and none of them are above 16%. The error could be attributed to landmark mistakes made by MediaPipe, the shortcomings of our algorithm, and a small number of people just perform the gesture differently from most people (*i.e.*, different from the ground truth we established before). The results show good generalizability since the parameters are only tuned on a small number of samples.

Table 3. Rule Parameters, and Their Performance on HaGRID Test Set

Rule	Parameters	Performance on HaGRID Test Set		
		error	unsure	correct
Flexion - thumb	(16, 38)	0.036	0.457	0.507
Flexion - other fingers	(57, 74)	0.019	0.049	0.932
Proximity	(0.024, 0.029)	0.031	0.067	0.902
Contact	(0.046, 0.055)	0.020	0.024	0.956
Pointing Direction - thumb	40	0.047	0.239	0.714
Palm Orientation	41	0.042	0.075	0.882
Overall	-	0.023 \pm 0.018	0.062 \pm 0.039	0.916 \pm 0.053

To evaluate if the parameters trained on third-person view dataset can adapt to first-person view images, we tested them on a first-person view gesture dataset EgoGesture. We choose 20 gesture

classes (fist, measure, zero, one, two, three, four, five, six, seven, eight, nine, ok, three2, C, thumb down, thumb right, thumb left, thumb backward, thumb forward) and label the ground truth for each class. Each gesture has around 250 testing samples. The results are shown in Table 4. (Thumb pointing direction is not evaluated on this dataset because there is no adequate gesture for testing. Only the “thumb down” gesture has a clearly downward pointing direction, yet the images are not pointing strictly downward.)

On this first-person view dataset, the error rate of the rules only increase from 2.3% to 6.3% (though the correct rate decrease by around 19% because more “unsure” are predicted), showing that our rules works across different views.

Table 4. Performance of Rules on EgoGesture Test Set

Rule	Performance on EgoGesture		
	error	unsure	correct
Flexion - thumb	0.046	0.532	0.422
Flexion - other fingers	0.060	0.218	0.722
Proximity	0.098	0.194	0.708
Contact	0.044	0.159	0.797
Pointing Direction - thumb	-	-	-
Palm Orientation	0.108	0.250	0.642
Overall	0.063 ± 0.043	0.216 ± 0.096	0.720 ± 0.123

More detailed error analysis of our rules’ performance on EgoGesture Dataset can be found in Table 5. Only those with error rate above 15% are shown.

Table 5. Analysis of Error Cases on EgoGesture Dataset

Rule	Gesture	Finger	Error Rate	Observed Reasons
Flexion - other fingers	seven	ring	0.171	MediaPipe's mistake for occluded fingers.
	C	ring	0.595	The finger in this gesture is slightly bent by nature, hard to predict precisely.
		pinky	0.360	Same as above.
	thumb down	index	0.177	MediaPipe's mistake for occluded fingers.
		ring	0.326	Same as above.
	three-2	ring	0.281	Same as above.
Proximity	C	middle-ring	0.255	The rule does not generalize very well.
	three	middle-ring	0.154	Same as above.
	four	index-middle	0.260	Landmarks mistake; some people perform it differently; the fingers are slightly separated by nature, hard to predict precisely, but it doesn't influence the recognition of the gesture very much.
	five	middle-ring	0.536	Same as above.
		ring-pinky	0.353	Same as above.
		index-middle	0.289	Same as above.
	ok	middle-ring	0.767	Same as above.
		ring-pinky	0.488	Same as above.
		middle-ring	0.404	Same as above.
Contact	nine	ring-pinky	0.578	Same as above.
		index-middle	0.244	Landmarks mistake.
	seven	thumb-ring	0.202	Some people perform it differently; MediaPipe's mistake for occluded fingers.
		thumb-pinky	0.151	Same as above.
	measure	thumb-index	0.198	Landmark mistake; in some gestures thumb and index finger are close so it is hard to discriminate.
	nine	thumb-index	0.191	Landmark mistake.
		thumb-pinky	0.244	MediaPipe's mistake for occluded fingers.
	thumb down	thumb-index	0.223	Landmark mistake.
	thumb back-ward	thumb-index	0.249	Same as above.
	thumb for-ward	thumb-index	0.186	Same as above.

A.3 Pseudocode for description rules

Algorithm 1 Flexion of a finger

```

1: procedure FLEXION(finger, thresholdLow, thresholdHigh)
2:   curl  $\leftarrow$  0
3:   if finger is thumb then
4:     curl  $\leftarrow$  Angle( $\overrightarrow{\text{MCP-IP}}$ ,  $\overrightarrow{\text{IP-TIP}}$ )
5:   else
6:     curl  $\leftarrow$  Angle( $\overrightarrow{\text{MCP-PIP}}$ ,  $\overrightarrow{\text{PIP-DIP}}$ )
7:     curl  $\leftarrow$  curl + Angle( $\overrightarrow{\text{PIP-DIP}}$ ,  $\overrightarrow{\text{DIP-TIP}}$ )
8:   end if
9:   if curl  $\leq$  thresholdLow then
10:    return straight
11:   else if curl  $\geq$  thresholdHigh then
12:    return bent
13:   else
14:    return unsure
15:   end if
16: end procedure

```

Algorithm 2 Proximity of two fingers

```

1: procedure PROXIMITY(finger1, finger2, thresholdLow, thresholdHigh)
2:   jointDis  $\leftarrow$  0
3:   polylineF1  $\leftarrow$  Polyline(finger1 PIP, finger1 DIP, finger1 TIP)
4:   polylineF2  $\leftarrow$  Polyline(finger2 PIP, finger2 DIP, finger2 TIP)
5:   distance1  $\leftarrow$  Distance(finger1 PIP, polylineF2)
6:   distance2  $\leftarrow$  Distance(finger2 PIP, polylineF1)
7:   jointDis  $\leftarrow$  jointDis + min(distance1, distance2)
8:   distance3  $\leftarrow$  Distance(finger1 DIP, polylineF2)
9:   distance4  $\leftarrow$  Distance(finger2 TIP, polylineF1)
10:  jointDis  $\leftarrow$  jointDis + min(distance3, distance4)
11:  jointDis  $\leftarrow$  jointDis/3
12:  if jointDis is less than thresholdLow then
13:    return adjacent
14:  else if jointDis is greater than thresholdHigh then
15:    return separated
16:  else
17:    return unsure
18:  end if
19: end procedure

```

Algorithm 3 Contact of two fingers

```

1: procedure CONTACT(finger1, finger2, thresholdLow, thresholdHigh)
2:   distance  $\leftarrow$  Distance(finger1 TIP, finger2 TIP)
3:   if distance  $\leq$  thresholdLow then
4:     return contact
5:   else if distance  $\geq$  thresholdHigh then
6:     return notcontact
7:   else
8:     return unsure
9:   end if
10: end procedure

```

Algorithm 4 Thumb Pointing Direction

```

1: procedure CONTACT(thumb, threshold)
2:   if thumb is not straight then
3:     return unsure
4:   else
5:     Thumb  $\leftarrow \overrightarrow{\text{MCP-TIP}}
6:     minAngle  $\leftarrow +\infty$ 
7:     ThumbDir  $\leftarrow \text{None}$ 
8:     for dir in [down, up] do
9:       angle  $\leftarrow \text{Angle}(\text{Thumb}, \text{dir})
10:      if angle is less than minAngle then
11:        minAngle  $\leftarrow \text{angle}$ 
12:        ThumbDir  $\leftarrow \text{dir}$ 
13:      end if
14:    end for
15:    if minAngle is greater than threshold then
16:      return unsure
17:    else
18:      return ThumbDir
19:    end if
20:  end if
21: end procedure$$ 
```

Algorithm 5 Palm orientation

```

1: procedure PALMORIENTATION(hand, threshold)
2:   PalmVec1  $\leftarrow \overrightarrow{\text{pinkyMCP, indexMCP}}$ 
3:   PalmVec2  $\leftarrow \overrightarrow{\text{wrist, middleMCP}}$ 
4:   if hand is left hand then
5:     PalmOriVec  $\leftarrow \text{PalmVec1} \times \text{PalmVec2}$ 
6:   else if hand is right hand then
7:     PalmOriVec  $\leftarrow \text{PalmVec2} \times \text{PalmVec1}$ 
8:   end if
9:   minAngle  $\leftarrow +\infty$ 
10:  PalmOri  $\leftarrow \text{None}$ 
11:  for dir in [right, left, down, up, outward, inward] do
12:    angle  $\leftarrow \text{Angle}(\text{PalmOriVec}, \text{dir})$ 
13:    if angle is less than minAngle then
14:      minAngle  $\leftarrow \text{angle}$ 
15:      PalmOri  $\leftarrow \text{dir}$ 
16:    end if
17:  end for
18:  if minAngle is greater than threshold then
19:    return unsure
20:  else
21:    return PalmOri
22:  end if
23: end procedure

```

A.4 Gesture State Matrix Details

The matrix generated by the rule-based module is subsequently interpreted by Gesture Description Agent's gesture summary description generation module, and the result is provided to the Gesture Inference Agent for further analysis. This matrix encapsulates hand pose and movement status across two distinct channels, each offering a different dimension of gesture representation. This pose-movement split is proven to promote Gesture Description Agent's performance, avoiding omitting important characteristics or overemphasizing certain aspect.

Channel 1: Hand Pose The first channel is a 2D array comprising 19 rows and T columns, where T denotes the number of time steps, with each step representing 0.2 seconds. The rows are indexed starting from 1 and detail the following aspects:

- Rows 1-5 correspond to finger flexion for the thumb, index, middle, ring, and pinky fingers, respectively. The values are encoded as 1 (straight), 0 (between straight and bent), and -1 (bent), describing the extent of finger flexion.
- Rows 6-8 represent finger proximity for adjacent finger pairs (index-middle, middle-ring, ring-pinky) with similar encoding scheme. The aim is to indicate how closely each finger is to its neighbor.
- Rows 9-12 detail thumb fingertip contact with the fingertips of the other fingers, again using similar value encoding.
- Row 13 specifies the pointing direction of the thumb, with 1 (upward), -1 (downward), and 0 (no specific direction or unknown when thumb is bent).
- Rows 14-19 are dedicated to palm orientation, indicating the direction the palm faces from the user's perspective. A specific orientation is marked by a single row set to 1 among these rows, representing left, right, down, up, inward, and outward directions. All rows equal to 0 means no specific direction can be identified.

Channel 2: Hand Movement The second channel consists of a 2D array with 2 or 3 rows (depending on whether we can extract hand position in 3d space or 2d space) and T columns. On each time step, the vector corresponds to the geometric center of the hand at this time:

- Row 1 tracks the horizontal position (0 for leftmost to 1 for rightmost), where increasing values suggest movement from left to right.
- Row 2 follows the vertical position (0 for bottommost to 1 for topmost), where increasing values suggest movement from down to up.

To gauge movement magnitude, the hand's width is also provided. For example, a hand width of 0.05 with a rightward movement of 0.05 in the array suggests a displacement of approximately one hand width.

This detailed matrix representation ensures a comprehensive and nuanced understanding of hand gestures, facilitating advanced processing and interpretation in gesture recognition systems.

B Detailed Experiment Setting

This section outlines the experimental setup for two experiments carried out within this research.

B.1 Experiment 1: Augmented Reality-Based Smart Home IoT Control



Fig. 11. The experimental platform utilized in the smart home scenario. (a) IoT device control interface, simulated using Unity. (b) User wearing the Hololens and performing gestures with the right hand for device control.

The first experiment focuses on controlling IoT devices within a smart home environment through augmented reality. The setup simulates a scenario where users interact with various home devices using gesture controls.

As illustrated in Figure 11, the experimental platform integrates an augmented reality interface for IoT device control within a home setting.

The experimental setup encompasses a variety of device functions and user tasks to mimic real-world interactions with a smart home environment. Details for this experiment are presented in Table 7 for device functions and Table 6 for tasks and external contexts. The smart home scenario encompasses a total of 18 functions across 5 devices, offering a comprehensive assessment of gesture-based control in an augmented reality context.

B.2 Experiment 2: Online Video Streaming on PC

The second experiment focuses on user interaction with online video content on a PC monitor. To ensure familiarity with the website's interface among participants, we selected a highly popular video website for the experiment. Figure 12 illustrates the setup of our experimental platform.

Detailed descriptions of the tasks, including the number of functions and external context information for the video streaming environment, are provided in Table 8. Additionally, the list of functions available for tasks 4, 5, and 6 is presented in Table 9, while the function list for the remaining tasks is shown in Table ???. The variation in the function list is attributed to task 3, which involves full-screening the video page, resulting in a reduced number of available functions.

C System Cost for LLM Use

Our system employs the OpenAI API⁴ gpt-4-1106-preview for experiments, which is one of the best performance LLM and achieving near-human-level common sense and reasoning. The cost of each run is determined by the total token count.

⁴<https://openai.com/pricing#language-models>

Table 6. Task Instructions and External Context in the Smart Home Scenario

Task Instruction	External Context
Unlock the Smart Carbinet.	[“It is 7:00 PM now.”, “The child lock on this cabinet supports fingerprint unlocking.”]
Increase the brightness of the light.	[“It is 7:05 PM now.”]
Show the next recipes on the smart screen.	[“It is 7:12 PM now.”]
Open the oven.	[“It is 7:14 PM now.”, “Recipe instructions: now you need to open the oven”]
Open the air cleaner.	[“It is 7:20 PM now.”, “The air purifier’s sensor detected that the current environment has heavy cooking fumes.”]
Set a timer on the smart screen.	[“It is 7:30 PM now.”, “Recipe instructions: now you need to cook on high heat for five minutes.”]
Switch input source of the smart screen to the smart bell.	[“It is 7:32 PM now.”, “The doorbell is ringing.”]
Make a phone call through the smart screen.	[“It is 7:33 PM now.”, “Just now, it was the deliveryman delivering goods; the owner of the goods is the user’s roommate, Mark.”]

Table 7. Device Functions in Smart Home Scenario (Totally 18 Functions in 5 Devices)

Device Name	Function Name
Light	On / Off
	Brightness Control
	Mode Switch (Task Lighting / Morning Lighting / Accent Lighting)
	Child Lock Activated / Deactivated
Smart Cabinet	Temperature Control
	Humidity Control
	On / Off
Smart Screen	Switch Recipes (Recipe 1 / Recipe 2 / Recipe 3)
	Switch Input Source (Smart Screen / Smart Doorbell / iPad Video)
	Phone Call
	Settable Timer
	On / Off
Oven	Temperature Control
	Self Cleaning On / Off
	Mode Switch (Bake Mode / Convection Roast / Bottom Heat Only / Keep Warm / Energy Efficiency)
	On/Off
Air Cleaner	Airflow Speed Control
	Mode Switch (Strong / Silent / Custom)

For each gesture, GestureGPT consumes an average of 38785 tokens ($SD = 10432$) for input and an average of 3443 tokens ($SD = 1339$) for output, spanning 6.08 rounds of conversation ($SD = 0.85$). This results in a cost of \$0.389 per gesture.



Fig. 12. The experimental platform utilized in the video streaming scenario. (a) A user was watching the video and performing gesture to control it. (b) and (c) are the video interface and the function distribution.

Table 8. Task Instructions, Function Numbers and External Context in Video Streaming Scenario

Task Instruction	Function Numbers	External Context
Turn up the volume.	66	[“It is 8:01 PM now.”]
Drag the progress bar forward.	66	[“It is 8:02 PM now.”, “The user has watched the earlier part of this video.”]
Enter full screen mode.	66	[“It is 8:04 PM now.”]
Pause the video.	17	[“It is 8:15 PM now.”, “Right now, the user’s phone is actively receiving an incoming call.”]
Resume the video.	17	[“It is 8:17 PM now.”, “The user hung up the phone”]
Exit full screen mode.	17	[“It is 8:48 PM now.”]
Like the video.	66	[“It is 8:49 PM now.”]
Go to the next episode.	66	[“It is 8:50 PM now.”]

Table 9. Video Scenario Function List in Task 4, 5, 6

ID	Name	ID	Name
0	VideoProgressBarUpdate	9	SelectEpisode
1	PlayPauseButton	10	ChangePlaybackSpeed
2	NextButton	11	SubtitleControl
3	SeekTimeUpdate	12	VolumeControl
4	ToggleDanmakuDisplay	13	VideoSettingsMenu
5	DanmakuToggle	14	PictureInPictureToggle
6	DanmuEtiquetteHint	15	ToggleFullscreen
7	SendMessageButton	16	VideoPlayArea
8	VideoQualitySelection		

Although costs are currently elevated due to the premium on model resources and extensive token requirements, we anticipate a reduction in expenses as LLM technology continues to evolve.

D Data Samples and Prompts Utilized in GestureGPT

GestureGPT utilizes a triple-agent collaborative architecture, with each agent guided by its own specific prompt to steer the behavior of the LLM. The gesture data in GestureGPT also undergoes

several format transformations. To facilitate better utilization by the community, we will make our prompts and some corresponding data samples publicly available. Due to the length of the system prompt, it has been archived in a GitHub repository. The full prompt can be accessed at https://github.com/studyzx/GestureGPT_ISS.

References

- [1] Deepak Akkil and Poika Isokoski. 2016. Gaze Augmentation in Egocentric Video Improves Awareness of Intention. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, San Jose California USA, 1573–1584. <https://doi.org/10.1145/2858036.2858127>
- [2] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. 2022. Constitutional AI: Harmlessness from AI Feedback. <http://arxiv.org/abs/2212.08073> arXiv:2212.08073 [cs].
- [3] Sigal Berman and Helman Stern. 2012. Sensors for Gesture Recognition Systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42, 3 (May 2012), 277–290. <https://doi.org/10.1109/TSMCC.2011.2161077> Conference Name: IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews).
- [4] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricu, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. 2023. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. <http://arxiv.org/abs/2307.15818> arXiv:2307.15818 [cs].
- [5] Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2023. ChatEval: Towards Better LLM-based Evaluators through Multi-Agent Debate. <http://arxiv.org/abs/2308.07201> arXiv:2308.07201 [cs].
- [6] Ishan Chatterjee, Robert Xiao, and Chris Harrison. 2015. Gaze+Gesture: Expressive, Precise and Targeted Free-Space Interactions. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*. ACM, Seattle Washington USA, 131–138. <https://doi.org/10.1145/2818346.2820752>
- [7] Weihao Chen, Chun Yu, Huadong Wang, Zheng Wang, Lichen Yang, Yukun Wang, Weinan Shi, and Yuanchun Shi. 2023. From Gap to Synergy: Enhancing Contextual Understanding through Human-Machine Collaboration in Personalized Systems. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23)*. Association for Computing Machinery, New York, NY, USA, 1–15. <https://doi.org/10.1145/3586183.3606741>
- [8] William Delamare, Chaklam Silpasuwanchai, Sayan Sarcar, Toshiaki Shiraki, and Xiangshi Ren. 2019. On Gesture Combination: An Exploration of a Solution to Augment Gesture Interaction. In *Proceedings of the 2019 ACM International Conference on Interactive Surfaces and Spaces*. ACM, Daejeon Republic of Korea, 135–146. <https://doi.org/10.1145/3343055.3359706>
- [9] 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. <http://arxiv.org/abs/2306.06070> arXiv:2306.06070 [cs].
- [10] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Thompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. 2023. PaLM-E: An Embodied Multimodal Language Model. <http://arxiv.org/abs/2303.03378> arXiv:2303.03378 [cs].
- [11] Afshan Ejaz, Maria Rahim, and Shakeel Ahmed Khoja. 2019. The Effect of Cognitive Load on Gesture Acceptability of Older Adults in Mobile Application. In *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*. IEEE, New York City, NY, USA, 0979–0986. <https://doi.org/10.1109/UEMCON47517.2019.8992970>
- [12] Euan Freeman, Stephen Brewster, and Vuokko Lantz. 2016. Do That, There: An Interaction Technique for Addressing In-Air Gesture Systems. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*.

- Association for Computing Machinery, New York, NY, USA, 2319–2331. <https://doi.org/10.1145/2858036.2858308>
- [13] Isabel O. Gallegos, Ryan A. Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K. Ahmed. 2024. Bias and Fairness in Large Language Models: A Survey. <http://arxiv.org/abs/2309.00770> arXiv:2309.00770 [cs].
- [14] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: Program-aided Language Models. <https://doi.org/10.48550/arXiv.2211.10435> arXiv:2211.10435 [cs].
- [15] Qi Gao, Zheng Ma, Quan Gu, Jiaofeng Li, and Zaifeng Gao. 2023. Working Memory Capacity for Gesture-Command Associations in Gestural Interaction. *International Journal of Human-Computer Interaction* 39, 15 (Sept. 2023), 3045–3056. <https://doi.org/10.1080/10447318.2022.2091213>
- [16] Xianliang Ge, Yunxian Pan, Sujie Wang, Linze Qian, Jingjia Yuan, Jie Xu, Nitish Thakor, and Yu Sun. 2023. Improving Intention Detection in Single-Trial Classification Through Fusion of EEG and Eye-Tracker Data. *IEEE Transactions on Human-Machine Systems* 53, 1 (Feb. 2023), 132–141. <https://doi.org/10.1109/THMS.2022.3225633>
- [17] Jun Gong, Zheer Xu, Qifan Guo, Teddy Seyed, Xiang 'Anthony' Chen, Xiaojun Bi, and Xing-Dong Yang. 2018. WrisText: One-handed Text Entry on Smartwatch using Wrist Gestures. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, Montreal QC Canada, 1–14. <https://doi.org/10.1145/3173574.3173755>
- [18] Jun Gong, Xing-Dong Yang, and Pourang Irani. 2016. WristWhirl: One-handed Continuous Smartwatch Input using Wrist Gestures. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. ACM, Tokyo Japan, 861–872. <https://doi.org/10.1145/2984511.2984563>
- [19] Jun Gong, Yang Zhang, Xia Zhou, and Xing-Dong Yang. 2017. Pyro: Thumb-Tip Gesture Recognition Using Pyroelectric Infrared Sensing. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. ACM, Québec City QC Canada, 553–563. <https://doi.org/10.1145/3126594.3126615>
- [20] Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. 2024. A Real-World WebAgent with Planning, Long Context Understanding, and Program Synthesis. <http://arxiv.org/abs/2307.12856> arXiv:2307.12856 [cs].
- [21] Izzeddin Gur, Ofir Nachum, Yingjie Miao, Mustafa Safdari, Austin Huang, Aakanksha Chowdhery, Sharan Narang, Noah Fiedel, and Aleksandra Faust. 2023. Understanding HTML with Large Language Models. <http://arxiv.org/abs/2210.03945> arXiv:2210.03945 [cs].
- [22] Fang Hu, Peng He, Songlin Xu, Yin Li, and Cheng Zhang. 2020. FingerTrak: Continuous 3D Hand Pose Tracking by Deep Learning Hand Silhouettes Captured by Miniature Thermal Cameras on Wrist. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 2 (June 2020), 1–24. <https://doi.org/10.1145/3397306>
- [23] Chien-Ming Huang, Sean Andrist, Allison Sauppé, and Bilge Mutlu. 2015. Using gaze patterns to predict task intent in collaboration. *Frontiers in Psychology* 6 (July 2015). <https://doi.org/10.3389/fpsyg.2015.01049>
- [24] Lihi Idan. 2022. A Network-Based, Multidisciplinary Approach to Intention Inference. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*. ACM, New Orleans LA USA, 1–7. <https://doi.org/10.1145/3491101.3519754>
- [25] Toshiya Isomoto, Shota Yamanaka, and Buntarou Shizuki. 2022. Dwell Selection with ML-based Intent Prediction Using Only Gaze Data. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6, 3, Article 120 (sep 2022), 21 pages. <https://doi.org/10.1145/3550301>
- [26] Dou Jian-peng, Li Hang, Pang Xiao-Ling, Zhang Chao-Ni, Yang Tian-Huai, and Jin Xian-Min. 2019. Research progress of quantum memory. *Acta Physica Sinica* (2019). <https://doi.org/10.7498/APS.68.20190039>
- [27] Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. LongLLM-Lingua: Accelerating and Enhancing LLMs in Long Context Scenarios via Prompt Compression. (2023). <https://doi.org/10.48550/ARXIV.2310.06839> Publisher: [object Object] Version Number: 1.
- [28] Shuo Jiang, Ling Li, Haipeng Xu, Junkai Xu, Guoying Gu, and Peter B. Shull. 2020. Stretchable e-Skin Patch for Gesture Recognition on the Back of the Hand. *IEEE Transactions on Industrial Electronics* 67, 1 (Jan. 2020), 647–657. <https://doi.org/10.1109/TIE.2019.2914621>
- [29] A. Just, Y. Rodriguez, and S. Marcel. 2006. *Biometric Person Recognition Based on 3D Face Recognition*. Research Report RR-06-73. Idiap Research Institute. <https://infoscience.epfl.ch/record/146075/files/just-idiap-rr-06-73.pdf>
- [30] Alexander Kapitanov, Karina Kvachiani, Alexander Nagaev, Roman Kraynov, and Andrei Makhliarchuk. 2024. HaGRID - HAnd Gesture Recognition Image Dataset. <http://arxiv.org/abs/2206.08219> arXiv:2206.08219 [cs].
- [31] Evan King, Haoxiang Yu, Sangsu Lee, and Christine Julien. 2023. "Get ready for a party": Exploring smarter smart spaces with help from large language models. <http://arxiv.org/abs/2303.14143> arXiv:2303.14143 [cs].
- [32] Evan King, Haoxiang Yu, Sangsu Lee, and Christine Julien. 2024. Sasha: Creative Goal-Oriented Reasoning in Smart Homes with Large Language Models. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 8, 1 (March 2024), 1–38. <https://doi.org/10.1145/3643505>
- [33] Fatemeh Koochaki and Laleh Najafizadeh. 2018. Predicting Intention Through Eye Gaze Patterns. In *2018 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. IEEE, Cleveland, OH, 1–4. <https://doi.org/10.1109/BIOCAS.2018.8584665>

- [34] Peiyu Li, Ney Renau-Ferrer, É. Anquetil, and Eric Jamet. 2012. Semi-customizable Gestural Commands Approach and Its Evaluation. *2012 International Conference on Frontiers in Handwriting Recognition* (2012), 473–478. <https://doi.org/10.1109/ICFHR.2012.267>
- [35] Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Zhaopeng Tu, and Shuming Shi. 2023. Encouraging Divergent Thinking in Large Language Models through Multi-Agent Debate. <http://arxiv.org/abs/2305.19118> arXiv:2305.19118 [cs].
- [36] Hao Lü and Yang Li. 2011. Gesture avatar: a technique for operating mobile user interfaces using gestures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Vancouver BC Canada, 207–216. <https://doi.org/10.1145/1978942.1978972>
- [37] Hao Lü and Yang Li. 2012. Gesture coder: a tool for programming multi-touch gestures by demonstration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. Association for Computing Machinery, New York, NY, USA, 2875–2884. <https://doi.org/10.1145/2207676.2208693>
- [38] Ruilong Ma, Jingyu Wang, Qi Qi, Xiang Yang, Haifeng Sun, Zirui Zhuang, and Jianxin Liao. 2023. Poster: PipeLLM: Pipeline LLM Inference on Heterogeneous Devices with Sequence Slicing. In *Proceedings of the ACM SIGCOMM 2023 Conference (ACM SIGCOMM '23)*. Association for Computing Machinery, New York, NY, USA, 1126–1128. <https://doi.org/10.1145/3603269.3610856>
- [39] Naveen Madapana and Juan Wachs. 2019. Database of Gesture Attributes: Zero Shot Learning for Gesture Recognition. In *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)*. IEEE, Lille, France, 1–8. <https://doi.org/10.1109/FG.2019.8756548>
- [40] Naveen Madapana and Juan P. Wachs. 2018. Hard Zero Shot Learning for Gesture Recognition. In *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, Beijing, 3574–3579. <https://doi.org/10.1109/ICPR.2018.8545869> zero shot.
- [41] George B. Mo, John J Dudley, and Per Ola Kristensson. 2021. Gesture Knitter: A Hand Gesture Design Tool for Head-Mounted Mixed Reality Applications. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. Association for Computing Machinery, New York, NY, USA, 1–13. <https://doi.org/10.1145/3411764.3445766>
- [42] José Antonio Montero and L. Enrique Sucar. 2006. Context-Based Gesture Recognition. In *Progress in Pattern Recognition, Image Analysis and Applications*, David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, José Francisco Martínez-Trinidad, Jesús Ariel Carrasco Ochoa, and Josef Kittler (Eds.). Vol. 4225. Springer Berlin Heidelberg, Berlin, Heidelberg, 764–773. http://link.springer.com/10.1007/11892755_79
- [43] Louis-Philippe Morency and Trevor Darrell. 2006. Head gesture recognition in intelligent interfaces: the role of context in improving recognition. In *Proceedings of the 11th international conference on Intelligent user interfaces*. ACM, Sydney Australia, 32–38. <https://doi.org/10.1145/1111449.1111464>
- [44] K. Murray and G. Häubl. 2010. Freedom of Choice, Ease of Use, and the Formation of Interface Preferences. *Behavioral Marketing ejournal* (2010). <https://doi.org/10.2139/ssrn.1698204>
- [45] Miguel A. Nacenta, Yemliha Kamber, Yizhou Qiang, and Per Ola Kristensson. 2013. Memorability of pre-designed and user-defined gesture sets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. Association for Computing Machinery, New York, NY, USA, 1099–1108. <https://doi.org/10.1145/2470654.2466142>
- [46] Donald A. Norman. 2010. Natural user interfaces are not natural. *Interactions* 17, 3 (May 2010), 6–10. <https://doi.org/10.1145/1744161.1744163>
- [47] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoní Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiro, Matt Knight, Daniel Kokotajlo, Łukasz

Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O’Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael Pokorny, Michelle Pokrass, Witchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cérion Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, C. J. Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. GPT-4 Technical Report. <http://arxiv.org/abs/2303.08774> arXiv:2303.08774 [cs].

- [48] Jeongeon Park, Bryan Min, Xiaojuan Ma, and Juho Kim. 2023. ChoiceMates: Supporting Unfamiliar Online Decision-Making with Multi-Agent Conversational Interactions. <https://doi.org/10.48550/arXiv.2310.01331> arXiv:2310.01331 [cs].
- [49] Joon Sung Park, Joseph O’Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative Agents: Interactive Simulacra of Human Behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. ACM, San Francisco CA USA, 1–22. <https://doi.org/10.1145/3586183.3606763>
- [50] V.I. Pavlovic, R. Sharma, and T.S. Huang. 1997. Visual interpretation of hand gestures for human-computer interaction: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19, 7 (July 1997), 677–695. <https://doi.org/10.1109/34.598226> Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.
- [51] Chen Qian, Xin Cong, Wei Liu, Cheng Yang, Weize Chen, Yusheng Su, Yufan Dang, Jiahao Li, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Communicative Agents for Software Development. <http://arxiv.org/abs/2307.07924> arXiv:2307.07924 [cs].
- [52] Dmitriy Rivkin, Francois Hogan, Amal Feriani, Abhisek Konar, Adam Sigal, Steve Liu, and Greg Dudek. 2024. SAGE: Smart home Agent with Grounded Execution. <http://arxiv.org/abs/2311.00772> arXiv:2311.00772 [cs].
- [53] Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. 2023. From Words to Watts: Benchmarking the Energy Costs of Large Language Model Inference. <http://arxiv.org/abs/2310.03003> arXiv:2310.03003 [cs].
- [54] Julia Schwarz, Charles Claudius Marais, Tommer Leyvand, Scott E. Hudson, and Jennifer Mankoff. 2014. Combining body pose, gaze, and gesture to determine intention to interact in vision-based interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Toronto Ontario Canada, 3443–3452. <https://doi.org/10.1145/2556288.2556989>
- [55] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. <http://arxiv.org/abs/2303.11366> arXiv:2303.11366 [cs].
- [56] Ronal Singh, Tim Miller, Joshua Newn, Liz Sonenberg, Eduardo Veloso, and Frank Vetere. 2018. Combining Planning with Gaze for Online Human Intention Recognition. (2018).
- [57] Maximilian Speicher and Michael Nebeling. 2018. GestureWiz: A Human-Powered Gesture Design Environment for User Interface Prototypes. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI ’18)*. Association for Computing Machinery, New York, NY, USA, 1–11. <https://doi.org/10.1145/3173574.3173681>
- [58] Dina Stiegemeier, Sabrina Bringeland, J. Kraus, and M. Baumann. 2022. User Experience of In-Vehicle Gesture Interaction: Exploring the Effect of Autonomy and Competence in a Mock-Up Experiment. *Proceedings of the 14th International Conference on Automotive User Interfaces and Interactive Vehicular Applications* (2022). <https://doi.org/10.1145/3543174.3546847>
- [59] Zhida Sun, Sitong Wang, Chengzhong Liu, and Xiaojuan Ma. 2022. Metaphoraction: Support Gesture-based Interaction Design with Metaphorical Meanings. *ACM Transactions on Computer-Human Interaction* 29, 5 (Oct. 2022), 1–33.

- <https://doi.org/10.1145/3511892> [meaning].
- [60] Zsolt Szalavári and Michael Gervautz. 1997. The personal interaction Panel—a Two-Handed interface for augmented reality. In *Computer graphics forum*, Vol. 16. Wiley Online Library, C335–C346.
 - [61] Yashar Talebirad and Amirhossein Nadiri. 2023. Multi-Agent Collaboration: Harnessing the Power of Intelligent LLM Agents. <http://arxiv.org/abs/2306.03314> arXiv:2306.03314 [cs].
 - [62] Eugene M. Taranta Ii, Thaddeus K. Simons, Rahul Sukthankar, and Joseph J. Laviola Jr. 2015. Exploring the Benefits of Context in 3D Gesture Recognition for Game-Based Virtual Environments. *ACM Transactions on Interactive Intelligent Systems* 5, 1 (March 2015), 1–34. <https://doi.org/10.1145/2656345>
 - [63] Jean Vanderdonckt and Éric Petit. 2019. G-Menu: A Keyword-by-Gesture Based Dynamic Menu Interface for Smartphones. In *Human-Computer Interaction. Recognition and Interaction Technologies: Thematic Area, HCI 2019, Held as Part of the 21st HCI International Conference, HCII 2019, Orlando, FL, USA, July 26–31, 2019, Proceedings, Part II* 21. Springer, 99–114.
 - [64] Radu-Daniel Vatavu. 2012. User-defined gestures for free-hand TV control. In *Proceedings of the 10th European Conference on Interactive TV and Video (EuroITV '12)*. Association for Computing Machinery, New York, NY, USA, 45–48. <https://doi.org/10.1145/2325616.2325626>
 - [65] Chan Wah Ng and Surendra Ranganath. 2002. Real-time gesture recognition system and application. *Image and Vision Computing* 20, 13 (Dec. 2002), 993–1007. [https://doi.org/10.1016/S0262-8856\(02\)00113-0](https://doi.org/10.1016/S0262-8856(02)00113-0)
 - [66] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. 2024. A Survey on Large Language Model based Autonomous Agents. <http://arxiv.org/abs/2308.11432> arXiv:2308.11432 [cs].
 - [67] Tianyi Wang, Xun Qian, Fengming He, Xiyun Hu, Yuanzhi Cao, and Karthik Ramani. 2021. GesturAR: An Authoring System for Creating Freehand Interactive Augmented Reality Applications. In *The 34th Annual ACM Symposium on User Interface Software and Technology (UIST '21)*. Association for Computing Machinery, New York, NY, USA, 552–567. <https://doi.org/10.1145/3472749.3474769>
 - [68] Zhenhailong Wang, Shaoguang Mao, Wenshan Wu, Tao Ge, Furu Wei, and Heng Ji. 2023. Unleashing the Emergent Cognitive Synergy in Large Language Models: A Task-Solving Agent through Multi-Persona Self-Collaboration. (2023). <https://doi.org/10.48550/ARXIV.2307.05300> Publisher: [object Object] Version Number: 4.
 - [69] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. <http://arxiv.org/abs/2201.11903> arXiv:2201.11903 [cs].
 - [70] Alan Wexelblat. 1995. An approach to natural gesture in virtual environments. *ACM Transactions on Computer-Human Interaction* 2, 3 (Sept. 1995), 179–200. <https://doi.org/10.1145/210079.210080>
 - [71] Jinting Wu, Yujia Zhang, and Xiaoguang Zhao. 2021. A Prototype-Based Generalized Zero-Shot Learning Framework for Hand Gesture Recognition. In *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, Milan, Italy, 3435–3442. <https://doi.org/10.1109/ICPR48806.2021.9412548>
 - [72] Haijun Xia, Michael Glueck, Michelle Annett, Michael Wang, and Daniel Wigdor. 2022. Iteratively Designing Gesture Vocabularies: A Survey and Analysis of Best Practices in the HCI Literature. *ACM Transactions on Computer-Human Interaction* 29, 4 (2022), 37:1–37:54. <https://doi.org/10.1145/3503537>
 - [73] Xuhai Xu, Jun Gong, Carolina Brum, Lilian Liang, Bongsu Suh, Shivam Kumar Gupta, Yash Agarwal, Laurence Lindsey, Runchang Kang, Behrooz Shahsavari, Tu Nguyen, Heriberto Nieto, Scott E Hudson, Charlie Maalouf, Jax Seyed Mousavi, and Gierad Laput. 2022. Enabling Hand Gesture Customization on Wrist-Worn Devices. In *CHI Conference on Human Factors in Computing Systems*. ACM, New Orleans LA USA, 1–19. <https://doi.org/10.1145/3491102.3501904>
 - [74] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 20744–20757. https://proceedings.neurips.cc/paper_files/paper/2022/file/82ad13ec01f9fe44c01cb91814fd7b8c-Paper-Conference.pdf
 - [75] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. <http://arxiv.org/abs/2210.03629> arXiv:2210.03629 [cs].
 - [76] Yiyu Yao. 2012. An Outline of a Theory of Three-Way Decisions. In *Rough Sets and Current Trends in Computing*, David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, JingTao Yao, Yan Yang, Roman Słowiński, Salvatore Greco, Huaxiong Li, Sushmita Mitra, and Lech Polkowski (Eds.). Vol. 7413. Springer Berlin Heidelberg, Berlin, Heidelberg, 1–17. https://doi.org/10.1007/978-3-642-32115-3_1 Series Title: Lecture Notes in Computer Science.
 - [77] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, and Matthias Grundmann. 2020. MediaPipe Hands: On-device Real-time Hand Tracking. <http://arxiv.org/abs/2006.10214> arXiv:2006.10214 [cs].

- [78] Qiang Zhang, Yuanqiao Lin, Yubin Lin, and Szymon Rusinkiewicz. 2023. UltraGlove: Hand Pose Estimation with Mems-Ultrasonic Sensors. <http://arxiv.org/abs/2306.12652> arXiv:2306.12652 [cs].
- [79] Xinghua Zhang, Bowen Yu, Haiyang Yu, Yangyu Lv, Tingwen Liu, Fei Huang, Hongbo Xu, and Yongbin Li. 2023. Wider and Deeper LLM Networks are Fairer LLM Evaluators. <http://arxiv.org/abs/2308.01862> arXiv:2308.01862 [cs].
- [80] Yifan Zhang, Congqi Cao, Jian Cheng, and Hanqing Lu. 2018. EgoGesture: A New Dataset and Benchmark for Egocentric Hand Gesture Recognition. *IEEE Transactions on Multimedia* 20, 5 (May 2018), 1038–1050. <https://doi.org/10.1109/TMM.2018.2808769>
- [81] Yu Zhang, Tao Gu, Chu Luo, Vassilis Kostakos, and Aruna Seneviratne. 2018. FinDroidHR: Smartwatch Gesture Input with Optical Heartrate Monitor. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 1 (2018), 56:1–56:42. <https://doi.org/10.1145/3191788>
- [82] Junchen Zhao, Yurun Song, Simeng Liu, Ian G. Harris, and Sangeetha Abdu Jyothi. 2023. LinguaLinked: A Distributed Large Language Model Inference System for Mobile Devices. <http://arxiv.org/abs/2312.00388> arXiv:2312.00388 [cs].
- [83] Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. 2024. GPT-4V(ision) is a Generalist Web Agent, if Grounded. <http://arxiv.org/abs/2401.01614> arXiv:2401.01614 [cs].
- [84] Qihuang Zhong, Liang Ding, Juhua Liu, Bo Du, and Dacheng Tao. 2023. Can ChatGPT Understand Too? A Comparative Study on ChatGPT and Fine-tuned BERT. <http://arxiv.org/abs/2302.10198> arXiv:2302.10198 [cs] version: 2.
- [85] Mingchen Zhuge, Haozhe Liu, Francesco Faccio, Dylan R. Ashley, Róbert Csordás, Anand Gopalakrishnan, Abdullah Hamdi, Hasan Abed Al Kader Hammoud, Vincent Herrmann, Kazuki Irie, Louis Kirsch, Bing Li, Guohao Li, Shuming Liu, Jinjie Mai, Piotr Piękos, Aditya Ramesh, Imanol Schlag, Weimin Shi, Aleksandar Stanić, Wenyi Wang, Yuhui Wang, Mengmeng Xu, Deng-Ping Fan, Bernard Ghanem, and Jürgen Schmidhuber. 2023. Mindstorms in Natural Language-Based Societies of Mind. <http://arxiv.org/abs/2305.17066> arXiv:2305.17066 [cs].

Received 2024-07-01; accepted 2024-09-20