GestureGPT: Zero-shot Interactive Gesture Understanding with Large Language Model Agents

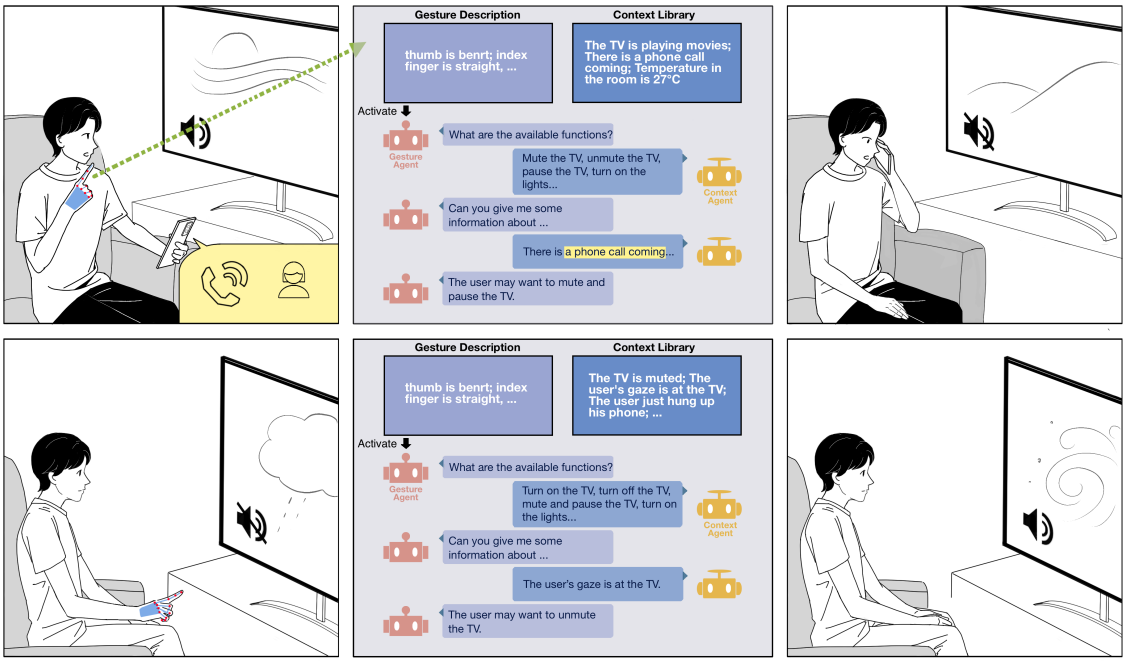

Fig. 1. Example interaction of GestureGPT. a) Tom is watching TV at home and he just receives a phone call from his mom. To create a more silent environment for communication, he performs a 'mute' gesture to indicate that he wants to mute the TV. b) The landmarks of the hand is extracted and used to generate rule-based descriptions and consequently activate *Gesture Agent*. Context information including states of devices, gaze position, interaction history, etc. Then, *Gesture Agent* and *Context Agent* have a multi-turn dialogue to predict the user's interaction intention. c) The TV is muted and Tom talks to his mom on the phone. d) After Tom hangs up the phone, Tom looks at the TV and makes a point gesture to indicate that he wants to continue watching TV. e) The gesture description and context information, including the user history that he muted the TV just now, is sent to *Gesture Agent* and *Context Agent* respectively. f) The TV is unmuted.

Current gesture recognition systems primarily focus on identifying gestures within a predefined set, leaving a gap in connecting these gestures to interactive GUI elements or system functions (e.g., linking a 'thumb-up' gesture to a 'like' button). We introduce

Manuscript submitted to ACM

GestureGPT, a novel zero-shot gesture understanding and grounding framework leveraging large language models (LLMs). Gesture descriptions are formulated based on hand landmark coordinates from gesture videos and fed into our dual-agent dialogue system. A gesture agent deciphers these descriptions and queries about the interaction context (e.g., interface, history, gaze data), which a context agent organizes and provides. Following iterative exchanges, the gesture agent discerns user intent, grounding it to an interactive function. We validated the gesture description module using public first-view and third-view gesture datasets and tested it in two real-world settings: video streaming and IoT control. The zero-shot Top-5 grounding accuracies are 52.3% for video streaming and 68.7% for IoT control tasks for user-defined free gestures, showing potential of the new gesture understanding paradigm.

CCS Concepts: • **Human-centered computing** → **User interface management systems**; • **Computing methodologies** → **Natural language processing**.

Additional Key Words and Phrases: zero-shot, gesture recognition, interaction context

ACM Reference Format:

. 2018. GestureGPT: Zero-shot Interactive Gesture Understanding with Large Language Model Agents. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 26 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Gestures offer an intuitive and immediate channel of expression of human intents, which is of vital importance for communication between humans and computers. They enable users to interact with digital systems in a manner that mirrors real-world interactions, reducing cognitive load and enhancing natural user experience. Thus, numerous efforts have gone into the development of gestural interaction interfaces. However, most of the work in this field focuses on gesture recognition tasks; *i.e.*, classify the gesture performed into a category within a predefined gesture set. On one hand, users need to learn the predefined gestures, which limits the spontaneity and expressiveness; on the other hand, the digital system still need to associate the gesture category with an interactive element or function, which is highly context-depended and not always straightforward. Such constrains of traditional gesture interfaces make them feel less natural and intuitive, which leads to Norman's famous argument that "Natural Interfaces are Not Natural" [ref]. While significant advances have been made in gesture recognition technologies over the years, the naturalness challenge persists: the translation of gestures into actual interaction actions.

As Norman pointed out, gestures are ephemeral and thus highly relevant to the interaction context, including user identity and culture background, interaction interface, interaction history, and other modalities of concurrent input (*e.g.*, gaze, speech). Recognizing the importance of interaction context for natural gesture interfaces, we propose GestureGPT, a context-aware gesture understanding and grounding framework that enables zero-shot association between gestures and interactive functions. GestureGPT leverages the prior knowledge of gestures and context information of large language models (LLM) to interpret the interaction intention behind the performed gesture. To that end, we introduce an dual-agent dialogue system that involves both an LLM *Gesture Agent* and an LLM *Context Agent*. To transfer gestures into natural language for LLM understanding, we design a set of rules to describe gestures based on hand landmarks detected on visual data. The state of each finger, the closeness of adjacent fingers, and fingers and hand moving directions (if any) are described according to the rules by calculation of angles and distances of finger joints. *Gesture Agent* takes the summarized description as input, inquires *Context Agent* about the context information, and predicts which interactive function the user intents to activate after synthesizing the gesture and context information. *Context Agent* organizes a context knowledge base and answers the questions of *Gesture Agent* accordingly. For each detected gesture, the two agents engage in iterative conversational exchanges for a holistic understanding of the interaction situation, which leads to a dynamic grounding of the gesture with a interactive function.

GestureGPT offers a suite of advantages that address the challenges and limitations of traditional gesture recognition systems. Firstly, its zero-shot gesture understanding capability allows for the recognition of gestures beyond predefined gesture sets, allowing more natural and intuitive gesture interfaces; Secondly, it goes beyond traditional gesture classification and actually grounds the gesture to a interactive function, bridging the gap between gesture recognition and actionable interface elements; This is achieved by novelly leveraging LLM's prior knowledge of gestures and contexts, which enables holistic understanding of the interaction gesture and context. Thirdly, GestureGPT relies on visual-based hand landmarks to recognize gestures, which is robust to changes of angle views thanks to existing advanced computer vision techniques. Rather than a static mapping of gestures, GestureGPT dynamically grounds them to interactive GUI elements, offering flexibility in interactions and addressing the ephemeral challenge of gestures.

We designed and evaluated our gesture description module on public datasets with first-view and third-view gesture images and videos. Our description rules have an overall description accuracy of xx% on xx test samples (xx static, yy dynamic gestures). The raw descriptions are summarized by a large language model before feeding to *Gesture Agent*. We conducted two user studies to evaluate our system in two typical interaction scenarios: a third-view video streaming on PC scenario and a first-view IoT device control with AR headset scenario. The overall Top-1, Top-3, and Top-5 accuracies are 45.7%, 54.7%, and 61.1% for user-defined free gestures and 44.4%, 56.1%, and 60.6% for designer-designated gestures averaged across all tasks of both scenarios.

Our contribution is three-fold:

(1) We proposed a novel dual-agent gesture understanding framework that grounds gestures to interactive functions in a zero-shot manner by leveraging large language models' prior knowledge of gestures and context, closing the gap between gesture recognition and system actions.
(2) We designed a set of gesture description rules based on hand landmarks extracted from images and videos, which is compatible with existing interactive and agnostic to angle of views.
(3) We evaluated the effects of different contexts on the gesture grounding task, providing guidance for future context-aware gesture understanding work.

2 BACKGROUND AND RELATED WORK

2.1 Context-aware Gesture Recognition

Gestures have long been a pivotal component in human-computer interactions, garnering significant attention in associated fields. As early as the previous century, researchers started introducing neural networks for hand gesture recognition[15]. Numerous advanced methodologies have seen substantial evolution over time [11, 14]. However, while these methods yield satisfactory performance, their practical applications are constrained by two factors: Firstly, high-performing algorithms depend on large-scale annotated data and struggle to adapt to diverse scenarios or user groups. Secondly, many gesture recognition methods pre-define a specific set of gesture categories and does not generalize to unseen gestures. This limitation is particularly problematic because many applications demand custom interaction semantics.

Accordingly, the focus has shifted towards recognizing gestures in low-resource or few-shot settings [13, 19]. Rahimian et al. [19] first explored this few-shot learning setting. A more recent contribution by Maslych et al. [13] significantly enhanced performance by effectively integrating multiple data augmentation strategies. However, although the need for large-scale annotation has been alleviated, these improved methods still necessitate the pre-definition of gesture categories. Contrarily, in immersive environments, hand gestures can encapsulate rich and diverse semantics

when adapt to different devices or contexts. For instance, [26] demonstrated that even nuanced gestures could be informative within an interactive framework. In an ideal setting, users should be able to spontaneously express fine-grained, adaptive gestures to communicate their interactive intentions. To this end, this paper innovatively leverages the prior knowledge of LLMs to enable adaptive comprehension of spontaneous user gestures in pervasive environments, empowering users to express their interaction intentions both intuitively and naturally.

Context provide informative clue for understanding users' interactive intent in pervasive environment. For example, Schilit et al. [21] from XEROX PARC uses location, distance, and time as context to trigger events. Dey et al. [6] formulates a conceptual framework for context-aware applications, and introduces the Context Toolkit to facilitate the creation of such applications. Dey et al. [7] propose programmable prototyping environment where users can demonstrate adaptive behaviour to enable context-aware application. Gu et al. [9] proposes an ontology-based context architecture to reason about various contexts. In our work, we use a large language model agent to manage and synthesize context information. Compared to previous work, LLM agent provides a fast access to context information through conversational interface, greatly reduces the complexity of context information extraction. It is even possible for LLM agent to abstract high-level context information from low-level context data like location and time, thanks to the reasoning capability and common sense of LLMs.

2.2 Large Language Model as Autonomous Agent

Large language models have displayed an exceptional ability to understand and execute a broad spectrum of natural language tasks[1, 16]. Numerous recent studies have delved into the feasibility of using LLMs beyond the confines of traditional natural language processing. These studies aim to emulate human-level intelligence, allowing LLMs to accurately perceive generalized environments and act accordingly when faced with diverse situations[23]. Agents based on LLMs show potential in various domains, from web browsing[5, 27] and strategic planning[22, 28] to robotic control[2, 8]. Deng et al. [5] use LLMs as web agents, establishing benchmarks for a multitude of web tasks spanning hundreds of websites across varied domains, such as booking flights or locating specific information. Yao et al. [28] introduced a prompting strategy in which LLMs reason and act by externalizing their thoughts and assimilating the corresponding observations. Subsequent research discovered that LLMs can self-identify errors, iterating improvements to enhance outcomes[22]. Brohan et al. [2] show that when grounded in visual environments, LLMs can interpret and execute robotic instructions, even with previously unseen objects and tasks. Finally, frameworks like AutoGPT[1] and LangChain[2] integrate various agents and hold promise as powerful personal assistants. Distinct from the aforementioned studies, this paper focus on the exploration of LLM-based agents to comprehend free-form user gestures intentions and execute corresponding instructions in pervasive computing environments.

Several recent works further explore the collaboration of multiple agents to accomplish more complex tasks. For instance, Park et al. [17] designed a multi-agent system that simulates human behavior in a virtual environment, where agents can perform daily tasks such as cooking breakfast or initiating conversations. Qian et al. [18] utilize multiple LLM agents for software development, enable communication and mutual verification, eventually producing a holistic software solution, inclusive of source codes, environment dependencies, and user manuals. Saha et al. [20] demonstrated how an LLM can teach a weaker language model, providing natural language explanations for reasoning skills through a theory of mind. Other applications span reasoning[12], evaluation[4, 30], and a myriad of intricate tasks[31]. In comparison, GestureGPT employs two separate agents: one for interpreting free-form gestures and

[1]https://github.com/Significant-Gravitas/Auto-GPT
[2]https://github.com/langchain-ai/langchain

another for environmental context management. These two agents collaborate seamlessly to achieve accurate gesture comprehension and execution.

3 METHOD

GestureGPT has a context-aware gesture understanding and grounding framework supporting zero-shot association of gestures with interactive functionalities. By synthesizing gesture descriptions and context information, GestureGPT discerns the intended interaction behind each gesture without additional training for the grounding task. The whole framework can be seen in fig 1

At the heart of this framework lies a novel dual-agent dialogue system. This system comprises a LLM-based **Gesture Agent** and a **Context Agent**, as well as a **Gesture Description** utility module. The gesture description module translates gesture images into natural language so that gestures become comprehensible for LLMs. We carefully crafted a set of rules that characterize gestures with the finger and joint states. These descriptions are extracted from hand landmarks derived from visual data, detailing each finger's status, their relative closeness, and their directional motion. These decisions rely on intricate calculations of angles and distances between finger joints. To manage uncertain cases, our rules adapt flexibly and only output results with high confidence, avoiding the generation of misleading information. Further details are provided in section 4.

Upon receiving the gesture descriptions, *Gesture Agent* consults *Context Agent* to comprehend the current context. After that, it synthesize the gesture and contextual data to infer the user's intended interactive function. Initially, *Gesture Agent* requests information about the interaction interface and the available target function list. Based on this data, it determines additional context information that can help the gesture grounding task, and seeks them from the *Context Agent*. The Gesture Agent progressively refines its understanding of the gesture with the evolving context until it can confidently predict the user's intention, *i.e.*, the target interactive function.

Context Agent manages a context library and acts as an conversational interface for the *Gesture Agent* to access different types of context information. Depending on its knowledge of the context library, *Context Agent* interprets and abstracts the context information, ensuring a fast and reliable response to queries from *Gesture Agent*.

The two agents engage in iterative dialogue for a thorough understanding of the interactive scenario. Their collaboration results in a dynamic association between the gesture and its corresponding interactive function.

4 RULE-BASED GESTURE DESCRIPTION MODULE

The gestures are described based on hand landmarks and a set of rules for finger states. Our system first extract 24 hand landmarks using mature computer-vision techniques, then calculate features like finger spread status and joint angle. We define a set of rules based on such features to describe gesture in terms of finger state, closeness to each other, etc. The description is later provided to the *Gesture Agent* for further synthesis.

4.1 Hand Landmarks Extraction

We use MediaPipe[29] to extract 3D hand landmarks from images. The 3D hand landmark consists of 21 hand-knuckle points (including hand joints and wrist) as shown in Fig. **??**. They are real-world 3D coordinates in meters with the origin at the hand's geometric center and robust across different angle-of-views, hand-camera distances, and other factors. This greatly improves the generalizability of our method to different interaction scenarios.

4.2 Gesture Description Rules

We define rules for both static gestures (*i.e.*, poser) and dynamic gestures (*i.e.*, involve fingers or hand movement). Such rules describe flexion of individual fingers, proximity and touch of different fingers, palm direction, as well as hand and finger moving directions if any. For each rule, a state of 'unsure' is added in the prediction, because incorrect descriptions can be greatly misleading to *Gesture Agent*. The rules are summarized in Table 1.

Table 1. Rules for Static Hand State Description

Rule Type	Rule Name	Applicable to	Value	Parameters
static	flexion	thumb	straight, bent, (unsure)	*straight_threshold*, *bent_threshold*
		index finger, middle finger, ring finger, pinky finger		
	proximity	index-middle finger, middle-ring finger, ring-pinky finger	pressed together, separated, (unsure)	*together_threshold*, *separated_threshold*
	touch	thumb-index finger, thumb-middle finger, thumb-ring finger, thumb-pinky finger	contact, not contact, (unsure)	*contact_threshold*, *not_contact_threshold*
	palm direction	hand	up, down, left, right, inward, outward, (unsure)	*angle_threshold*
dynamic	hand moving direction	hand	static, up, down, left, right, inward, outward, (unsure)	*moving_threshold*
	finger moving direction	thumb	static, inward, outward, (static)	*moving_threshold*
		index finger, middle finger, ring finger, pinky finger		

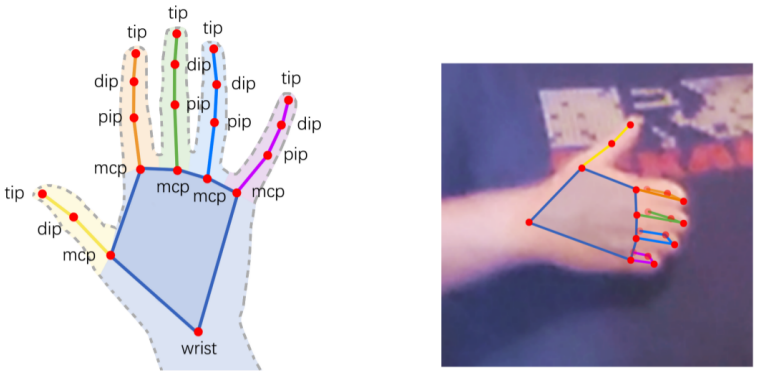

Fig. 2. A typical gesture's landmark and expected descriptions based on the rules: a) A picture of 'like' gesture clipped from the HaGRID Dataset; b) The extracted landmark by MediaPipe. Expected descriptions obtained by the rules is: Thumb is straight, index finger, middle finger, ring finger and pinky finger are bent. Index finger, middle finger, ring finger and pinky finger are pressed together. Thumb's fingertip is in contact with the other four fingers. Palm is facing outward. (Not literally the same as the generated description. For simplicity, we just keep the meaning reserved.)

4.2.1 Rules Definition for Static Gestures. For static gestures, we designed a set of rules to describe the state of the hand and each finger, including

Flexion Whether the finger is straight or bent (applicable to all fingers).

Proximity If two fingers are pressed together or separated (only applicable to index-middle finger, middle-ring finger, ring-pinky finger).

Touch If thumb's fingertip is in contact with other four fingers' fingertip.

Palm Direction The direction of the palm including upward, downward, left, right, forward, backward.

4.2.2 Additional Rules Definition for Dynamic Gestures. Aside from the above rules, there are two dynamic rules to describe the moving characteristics of the hand and each finger:

Hand Moving State The moving state of the hand including static, upward, downward, left, right, forward, backward.

Finger Moving State If a finger is static, moving inward or outward relative to the palm.

4.2.3 Rules Calculation Method. **Flexion** of a finger is computed 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 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 A and B is computed 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.

Touch of two fingers 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.

Palm direction is computed as the direction to which the palm is facing, which is obtained from the cross product of two vectors from hand (Fig. 3). 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 direction vector, and the angle is below *angle_threshold*, the reference vector would be the palm direction. Else the direction of palm is set as 'unsure'.

Hand Moving State of a dynamic gesture is computed as the moving direction and distance of the hand. First, the moving vector is calculated as the moving of the geometrical center of a hand landmark, from the first frame to the last frame. If the vector's length is smaller than *moving_threshold*, it is recognized as 'static'. Else the vector is compared with some reference vectors, and the reference vector which has the minimal angle with the moving vector is recognized as the moving direction.

Finger Moving State of a finger in dynamic gesture is computed as the change of angle formed by wrist, mcp and tip, from the first frame to the last frame. If it's smaller than *moving_threshold*, the finger is recognized as static. Else, whether the finger is moving inward or outward is decided by the changing direction of the angle.

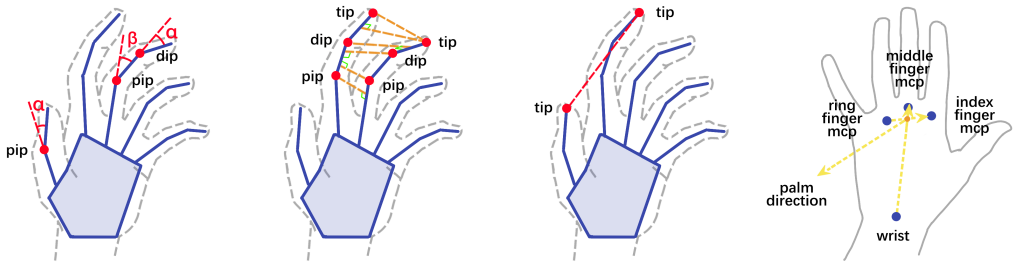

Fig. 3. Calculation of static rules. a) The flexion of a finger is calculated as the sum of bending angle of ip joint (for thumb) or pip and dip joint (for other fingers). b) The proximity of two fingers is calculated as the average distance from each finger's pip/dip/tip joint to the other finger. c) the touch of thumb and another finger is calculated as the distance of two finger's fingertips. d) The palm direction is calculated from the dot product of the two vectors on the hand, pointing towards the reader.

4.3 Parameter Setting and Evaluation Results

For there is no dataset that dynamic gestures are segmented precisely, which makes it difficult to tune a effective result, we only tune the parameters for static gestures. The dynamic rules' parameters are derived from those of static rules.

To determine the parameters (i.e. the thresholds) in the static rules mentioned above, we used HaGRID[10] dataset to tune the algorithm. HaGRID contains pictures of people performing 18 kinds of static gestures. For each gesture, we assume that most people perform it in the same way. Thus, when applying the rules to a set of gesture pictures of the same kind, most results would be the same. However, ambient exceptions like 'whether thumb is straight or bent in the peace gesture' exist, in which both answers are acceptable. Cases like this are excluded from the parameters setting process. The ground truth of each gesture when applying each rule is initially established by the common knowledge of the gesture. If needed, a gesture can have more than one ground truth label (and it would not engage in the training and testing process).

A subsample of HaGRID is used to tune the thresholds which has 100 pictures for each gesture, and 1800 pictures in total. In our implementation, a small number of samples are filtered out because MediaPipe failed to extract landmarks of the correct hand. Finally we have 1524 pictures, 70 to 96 pictures for each gesture. For most rules except flexion on thumb, we use the whole subsample to tune the parameters. For flexion on thumb, due to the severe imbalance of samples with labels 'straight' and 'bent' respectively, we specially assign the training set, which is composed of gesture 'like' (thumb is straight) and gesture 'ok' (thumb is bent, manully filtered to ensure the quality of the training data).

A grid search is conducted to find out the best parameter with minimal loss, which is a variation of accuracy. Because the existence of 'unsure' in outputs, we need to treat it differently from the cases when the algorithm outputs an actually incorrect prediction. The test results are divided into three circumstances: 1) Error: incorrect prediction (misleading information); 2) Unsure: no mention of the rule with 'unsure' state; 3) Correct: correct prediction (useful information).

The tuning results and performance on the HaGRID test set (38576 images) is shown in table 2. For most rules, the average error rate among all gestures is below 4% and the overall accuracy is above 89%. One exception is flexion rule for thumb, in which the output 'unsure' takes about 50% 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 training and testing. By adding 'unsure', we avoid the misleading information that potentially confuses the *Gesture Agent*.

Error rates for most gestures are below 5% and none of them are above 16%. The error could originate from landmark mistake made by MediaPipe, the shortcomings of our algorithm, and a small number of people just perform the gesture in a way different from most people (i.e. different from the ground truth we established before). The results show good generalization ability since the parameters are only tuned on a small number of samples.

Table 2. Parameters and Performance of static rules 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
Touch	(0.046, 0.055)	0.020	0.024	0.956
Direction of the palm	41	0.038	0.067	0.895

To evaluate if the parameters trained on third-view dataset have good performance for first-view images, we tested them on a first-view gesture dataset EgoGesture. We choose 20 static gestures (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 truths. For each gesture, we selected around 100 testing samples. The results are shown in Table 3.

Table 3. Performance of static rules on EgoGesture Test Set

Rule	**Performance on EgoGesture**		
	error	**unsure**	**correct**
Flexion - thumb	0.053	0.449	0.499
Flexion - other fingers	0.071	0.104	0.825
Proximity	0.129	0.096	0.775
Touch	0.067	0.050	0.883
Palm Direction	0.184	0.186	0.631

More detailed error analysis of our gesture description module's performance can be found in Appendix A. The performance of the algorithm on the first-view dataset is only slightly decreased, showing that our gesture describing algorithm works across different views.

For dynamic rules, we derive the parameters from those of static rules. For finger moving direction, the *moving_threshold* is the difference of *straight_threshold* and *bent_threshold* in flexion. For hand moving direction, the parameter *moving_threshold* is set to 0.1 as an experience value.

5 DUEL-AGENT SYSTEM DESIGN

GestureGPT takes the dual-agent conversational framework where each agent has its own specific roles. The primary challenge lies in guiding two general-purpose LLM models with delicately designed prompting methods to excel in the untrained domain of gesture interaction, particularly when intertwined with environmental context.

LLMs can understand complex instructions and act as designated agents, but only when provided with concrete, structured prompts and augmented with appropriate reasoning logic. Many recent works have made significant progress and efforts along the way[17, 22, 24, 25, 28]. There are generally several important design principles for prompting LLMs, for example, write clear and structured instructions *TODO_XIN: ref 1*, externalize the reasoning process *TODO_XIN: ref 2*, etc. In the following, we explain our prompting methods respectively for the gesture agent and context agent in detail, and demonstrate that such a structured thought process could guide the LLM to dissect the problem in a human-analogous analytical manner.

5.1 Gesture Agent Design and Implementation

Gesture agent first receives the description of any spontaneous gestures from user, it then reasons upon known situation and make necessary inquires for context environment in conversational exchange until it is confident to identify the user interaction intent.

We first define the role of gesture agent with specific and detailed description, this identity information is inserted in the *system slot* of LLM and will guide it be behave accordingly. The prompt is composed of four aspects as follows:

(1) **Agent Definition**: We begin by outlining the gesture agent's purpose, detailing its expected input, output, and conversational style. We specifically emphasize that the agent should organize its reasoning process step by step to emulate human-like cognitive processes. As identified by many existing work that such a procedure greatly improves the agent's performance.

Agent Definition

```
You are a gesture understanding assistant.
- Task Introduction
Your task is to predict the target interactive function from a function list based on a gesture
description and interaction context.
The gesture description will be provided to you directly, while you need to inquiry about the
context from a context agent.
You need to think step by step.
1. Try to understand the gesture based on the gesture description and have several guesses of what
the gesture is. PS: With the new information you have acquired, you can narrow down the candidates
based on interactive context information you gathered later.
2. Ask for the function list from the context agent.
3. Based on your understanding of the described gesture, ask for more specified context information
that you deem will help determine the target function. You can ask multiple times if necessary.
4. Synthesize the gesture and context information to better understand the interaction scenario.
5. Based on the available context, reexamine if your understanding of the gesture is accurate.
Restart from the first step if necessary.
6. Output the top-5 most possible functions based on your predictions of the gesture candidates and
the interactive context.
```

(2) **Gesture Description Rule**: We then clarify how the gesture description is created to help the agent better comprehend its potential meanings and discern interaction subtleties. We also include an illustration example to demonstrate these rules. A vital capability for the gesture agent is navigating ambiguous scenarios, often arising from obscured hand landmarks due to camera issues or finger occlusions. In these situations, the agent should actively make relevant contextual inquiries to summarize the interaction intent.

Gesture Description Rule

```
- Gesture Description Rule
You will receive a description about the gesture made by the user, including four static features
and two dynamic features, and all mentioned direction are from the user's first-person point of view:
1. whether a finger is straight or bent, and the straight finger's direction;
2. whether two fingers are pressed together or separated;
3. whether two fingers' fingertips are contact;
4. the palm's direction;
5. the finger's moving direction;
6. the hand's moving direction;
Not all gestures have the six aspects of description.
If the certain features are not mentioned at all or it's features is "unsure", you can guess their
states based on your understanding of similar gestures and the current interaction context that you
know of.
[Example Gesture Description for 'One' Gesture]
1. index finger is straight, middle finger is bent, ring finger is bent, pinky finger is bent;
2. index finger and middle finger are separated, middle finger and ring finger are pressed together,
ring finger and pinky finger are pressed together;
3. thumb and index finger's fingertips are not in contact, thumb and middle finger's fingertips are
in contact, thumb and ring finger's fingertips are not in contact, thumb and pinky finger's
fingertips are not in contact;
4. the palm is facing outward;
```

(3) **In-Context Demonstrations**: To further enhance the agent's capability, we provide a succinct conversation example as in-context demonstrations[3]. This ensures the agent responds appropriately during the conversation without overburdening the response with excessive analysis, which not only disrupt the context agent's behavior, causing confusion, but also notably increase the conversation token length, increasing the computational expense especially for commercial APIs.

In-Context Demonstration

```
- The Example of the Conversation
#You
(You must start with this question)
Can you provide the function list of the current interaction context? What kind of [type]
information can you provide me with?
#Context agent
Now interface have function
- expand information
- copy information
- add to cart
- show product 1
- show product 2
- show product 3
- show product 4
- show product 5
#You
(You can ask for whatever information you want, but you must specify the kind of information.) Can
you provide XXX information for me?
#Context agent
The user now XXXXXXXXX.
#You
(...You can ask questions to request the specific information you need. Please note that your
question should only contain the information you need; it should not include your analysis of the
gesture or the current interface.)
#Context agent
(... ...)
(... After some time conversing, if you feel you have enough information to make a decision, or the
conversation is no longer yielding new information ...)
#You
[Answer]
1. show product 2
2. show product 1
3. show product 3
4. copy information
5. expand information
[End]
[Reason]
(Explain the reasoning behind your answer.) .....
```

(4) **Task Termination And Explanation**: We finally formulate the expected output of the agent. Once it believes it has sufficient information to draw a conclusion, it is instructed to end the conversation, and give its top-5 most possible candidates followed by an explanation for the decisions. We empirically find the explanation aids in result analysis, offering clarity on the importance of context in the ongoing conversation.

Task Termination And Explanation

```
Please note that your question should only contain the information you need; it should not include
your analysis of the gesture or the current interface. All analyses should be reserved for the final
conversation under the "[Reason]" tag.

Your final answer should consist of two parts: the answer and the reason. The answer must include
exactly 5 options from the interface function list, arranged in descending order according to
probability; do not include any options not found on the function list from the context list. The
reason part should explain the rationale behind your chosen answer.

You answer should always contains the tag "[Answer]""[End]" and "[Reason]".
```

Based on the above instruction, the gesture agent initiates the conversation accordingly when receiving a new gesture description.

5.2 Context Agent Design and Implementation

The *Context Agent* is responsible for context management and responding to queries from the *Gesture Agent*, which seeks context information. Essentially, the *Context Agent* acts as a supervisor overseeing all sensors and other context sources. Thus, it functions as a manager rather than a sensor.

Similar to the gesture agent, the design of the context agent commences with a role description, which is divided into three main sections:

(1) **Agent Definition**: Beyond specifying the context task, it's vital to inform the context agent about the types of contexts it will encounter during interactions. In the current gestureGPT implementation, we recognize three primary context types:

- **Interface Description and its Value**: This is a JSON structure detailing the interface name (e.g., web of video playing, your home living room, etc.), the subarea of the interface (a categorization of the interface where each subarea encompasses specific functions), the functions within the interface, and their respective positions. It's worth noting that since 'how to summarize the context interface description' is out of our scope, we adopted the Wizard-of-Oz method, asking users to summarize the interface, which led to the approach we currently use.
- **User Gaze Information (Optional)**: This consists of a list of coordinates on the interface, representing the duration and location of the user's gaze during a gesture. Currently, we empirically select the gaze data from 1 second before the gesture starts until its conclusion.
- **Interaction History (Optional)**: This captures the user's interaction history over a specific period. At present, we consider the tasks leading up to the current moment as the interaction history.
- **High-Level Interaction Information (Optional)**: This isn't tied to a specific context category, meaning it can encompass information beyond the previous three types. As of now, we empirically include contexts such as environmental status (e.g., air quality sensor values, coming ring. the all high-level context used in our evaluation can be seen in User study)

Agent Definition

```
You are an interactive context assistant.
- Task Introduction
You are responsible for managing the current interactive context including interaction interface
understanding, interaction history memory, gaze location on the interface, etc.
You need to answer the questions based on your Context Knowledge Library.
Your context information library may contain some or all of the following items:
1. interface description: It contains divisions into subareas, and sub-subareas which having a
specified list of functions.
2. interface status.
3. user gaze position. When you providing gaze information, besides the coordinates of the gaze, you
need to combine it with the interface description to form a summary. You should tell the gesture
agent, the gaze data unavoidably contains certain degree level of inaccuracy.
4. interaction history.
5. other context information.
```

(2) **In-Context Demonstrations**: For the context agent, a crucial aspect of the example is to emphasize that it should refrain from responding if uncertain. This is to counteract a common issue with LLMs: Hallucination. We mandate that its responses must be grounded in the available context.

In-Context Demonstrations

```
- The Example of the Conversation
The following is an example conversation between you and a gesture understanding agent:
$$$ Current knowledge library:
interface function list;
interface status;
user gaze;
$$$
#Gesture agent
what interaction function list in this interface? what type context you can provide?
#You
Now interface have function
- expand information
- copy information
- add to cart
- show product 1
- show product 2
- show product 3
I can provide the following types of context:
- XXX
- XXX
...
#Gesture agent
I want know the user's now gaze information.
#You
(... Your answers should be based on the knowledge you possess; you should not fabricate facts from
thin air or exceed the scope of the question.)
(... When you providing gaze information, besides the coordinates of the gaze, you need to combine
it with the interface description to form a summary. You should tell the gesture agent, the gaze
data unavoidably contains certain degree level of inaccuracy. Like: Given that the user's gaze is
currently located at the coordinates XXX, which is within the xxx Area)
#Gesture agent
....
(After several rounds of conversation.)
#You
(... You can summarize the history of your conversation. If you have already provided all the
information you know, then you can respond, 'All known information has been shared; no additional
information can be provided.' or other same meaning ...)
```

(3) **Context Library**: This section houses the entirety of the current accessible context. To simplify implementation, we directly serialize the context into a JSON or list format for the prompt. Given our current use cases, the context library's length is finite, typically ranging between 20 to 1500 tokens

Context Library

```
Do not analyze the gesture information transmitted by the Gesture agent; Just provide the
information requested that is available in the context library.

As in the example, your task is to answer questions using only the information in your knowledge
base, without making additional inferences. If a question goes beyond the scope of your knowledge
base, you can respond, 'Based on my current knowledge base, I do not have access to this information.
```

Upon establishing its role, the context agent awaits queries from the gesture agent and responds based on its context library.

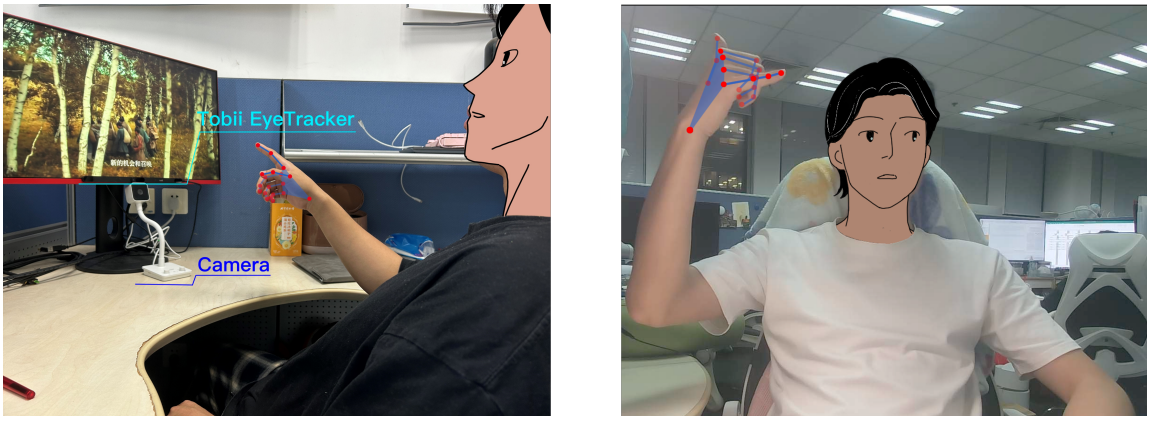

Fig. 4. The user sits at desktop and interact with computers using gestures.

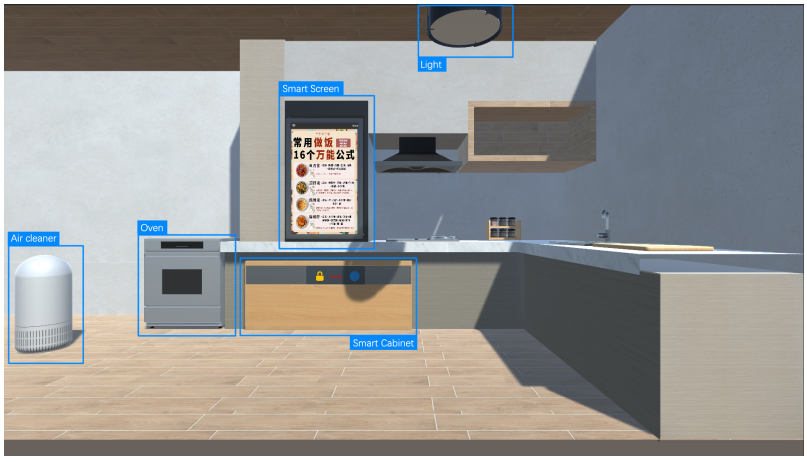

Fig. 5. Experiment setting for IoT device control scenario.

6 EVALUATION

We conducted two typical interaction scenarios: one involving third-view video streaming on a PC and the other involving first-view IoT device control using an AR headset.

Participants were recruited from three local schools, and they were compensated at a rate of $10 per hour. We informed participants that both video (including facial footage) and eye movement data would be collected during the study, and we assured them of the confidentiality and safety of their data. All participants provided informed consent. The participants in the two user studies were not totally identical.

In all user studies, we mandated that gestures be performed using only the right hand. Both dynamic and static gestures were permitted.

Each user study consisted of two sessions. The first session, termed the "freedom session," allowed participants to perform any gesture they believed best represented their intent for the given task. *TODO_XIN: Determine the total number of freedom gesture.* The second session, the "assigned session," required participants to perform specific gestures as instructed. At the end of the study, participants were asked to complete a questionnaire. This questionnaire utilized a 5-point Likert scale to evaluate the assigned gestures, with higher scores indicating an assigned gesture that better aligned with the user's intent.

6.1 User Study 1: Smart Home Control with AR Headset

In this scenario, users simulate cooking in a kitchen equipped with smart home devices that can be controlled through gestures. When users perform specific gestures, the status of the devices changes.

6.1.1 User Study Setting and Procedure. We used the HoloLens 2 headset as our experimental platform. This headset provides an API that allows us to obtain the user's gaze coordinates and hand gesture landmark coordinates directly. Additionally, we employed a separate USB camera to record the entire experimental process from a third-person view. Our platform was developed using Unity version 2020.3.24f1, MRTK 2.8.0, and the OpenXR Plugin 1.7.0. To enhance user immersion, we employed 3D models to represent the smart devices. The devices in this scene are a light, a smart cabinet, a smart screen, an oven and an air cleaner. The Experiment setting for IoT device control scenario can be seen in fig 5. The actual device control is facilitated using the "Wizard of Oz" rather than gesture recognition.

Our user study comprised 6 participants, with ages ranging from 23 to 26 (MEAN = 24.14, SD = 1.45). Among them, 4 were males and 2 were females.

Details regarding the tasks and their associated gestures are provided in Table 4.

TODO_XIN: add a device list and there function

6.1.2 Main Results Analysis. For our system evaluation, we employed two of OpenAI's leading APIs: gpt3.5 and gpt4, along with the highly popular open-source chat model, Vicuna-13B. The primary results presented in this section are derived from gpt4 due to its superior performance, results from the other models will be discussed in 7 *TODO: add the other model results in discussion section*.

The main results can be find in Table 5. We calculate the top1 rate, top3 rate, top5 rate(wich equal to the accuracy) and the negative rate(which equal to the 1-accuracy).

As we can see, the GestureGPT preform a supersing performance in this scenario. The most high accuracy was the 90.78% which show in the free gesture session. And the assigned session is 84.54%. For this results we can see a supuring findings that the free gesture session are better than assigned gesture session which we anticipate it should be

Table 4. Tasks, High Level Context and Assigned Gesture(in 'Assigned Session') in Smart Home Environment

Instruction	**Assigned Gesture**	**Additional Context**
Unlock the Smart Carbinet.	thumb up	The child lock on this cabinet supports fingerprint unlocking.
Increase the brightness of the light.	pinch and slide up (dynamic)	-
Show the next recipes on the smart screen.	pinch and slide left (dynamic)	-
Open the oven.	zooming in with full hand (dynamic)	Recipe instructions: now you need to open the oven.
Open the air cleaner.	zooming in with full hand (dynamic)	The air purifier's sensor detected that the current environment has heavy cooking fumes.
Set a timer on the smart screen.	five	Recipe instructions: now you need cook on high heat for five minutes.
Switch input source of the smart screen to the smart bell.	swiping left with full hand	The doorbell rang.
Make a phone call throught the smart screen.	call	Just now, it was the deliveryman delivering goods; the owner of the goods is the user's roommate, Mark.

better because of it's gesture designed by more experience HCI research and all user think this assigned gesutre are natural (from questionar, the mean point from 5-point Likert is 4.25(std=0.92). As we can see, the gaze information is very important for this scenario. beacese in smart

Table 5. All results For **Smart Home** with GPT4

	Free gesture				Assigned gesture			
	Top 1	Top 3	Top 5	Negative	Top 1	Top 3	Top 5	Nega
No Additional Context	$10.72\%_{\pm 1.74}$	$27.18\%_{\pm 3.93}$	$47.93\%_{\pm 7.81}$	$52.07\%_{\pm 7.81}$	$27.28\%_{\pm 3.41}$	$51.08\%_{\pm 5.20}$	$61.57\%_{\pm 4.02}$	$38.43\%_{\pm$
+Gaze	$52.48\%_{\pm 1.00}$	$71.63\%_{\pm 3.62}$	$90.78\%_{\pm 2.65}$	$9.22\%_{\pm 2.65}$	$45.83\%_{\pm 2.95}$	$68.75\%_{\pm 0.00}$	$83.33\%_{\pm 2.95}$	$16.67\%_{\pm$
+History	$13.48\%_{\pm 2.65}$	$27.66\%_{\pm 7.96}$	$52.48\%_{\pm 6.58}$	$47.52\%_{\pm 6.58}$	$24.31\%_{\pm 6.87}$	$51.39\%_{\pm 4.28}$	$64.58\%_{\pm 6.13}$	$35.42\%_{\pm$
+History+HLContext	$27.27\%_{\pm 6.83}$	$44.56\%_{\pm 7.04}$	$58.94\%_{\pm 5.04}$	$41.06\%_{\pm 5.04}$	$36.11\%_{\pm 6.44}$	$53.47\%_{\pm 2.60}$	$65.28\%_{\pm 5.47}$	$34.03\%_{\pm$
+Gaze+History	$46.81\%_{\pm 3.47}$	$67.38\%_{\pm 2.01}$	$87.23\%_{\pm 1.74}$	$12.77\%_{\pm 1.74}$	$41.55\%_{\pm 3.41}$	$64.76\%_{\pm 2.83}$	$84.45\%_{\pm 2.96}$	$15.55\%_{\pm$
+Gaze+History+HLContext	$56.03\%_{\pm 4.37}$	$73.76\%_{\pm 3.62}$	$87.94\%_{\pm 4.37}$	$12.06\%_{\pm 4.37}$	$47.16\%_{\pm 5.21}$	$68.39\%_{\pm 4.67}$	$84.54\%_{\pm 2.41}$	$15.46\%_{\pm$

6.1.3 Compared With Human Performance. For the human baseline, we enlisted 4 participants unfamiliar with our objectives. They were shown user gaze videos or gesture descriptions, in conjunction with gesture videos. Half of the participants received supplementary high-level context, while the rest did not. They were then tasked with identifying the top five functions that, in their estimation, the user intended to execute. This data was used to establish human performance benchmarks.

A comprehensive comparison across different settings can be viewed in Figure 6.

Table 6. Human Performance Result

	Free gesture				Assigned gesture			
	Top 1	Top 3	Top 5	Negative	Top 1	Top 3	Top 5	Negative
Home Scene Without HLContext(Human)	43.8%	62.5%	81.3%	18.7%	50%	75%	81.3%	18.7%
Home Scene With HLContext(Human)	58.3%	83.3%	91.7%	8.3%	91.7%	91.7%	100%	0%
Home Scene Without HLContext(GPT4)	41.67%	58.33%	79.17%	20.83%	29.17%	60.42%	75.00%	25.00%
Home Scene With HLContext(GPT4)	47.92%	64.58%	77.08%	22.92%	39.58%	60.42%	70.83%	29.17%
Video Scene Without HLContext(Human)	18.8%	43.8%	50%	50%	40.9%	47.8%	56.2%	43.8%
Video Scene With HLContext(Human)	83.3%	83.3%	100%	0%	100%	100%	100%	0%
Video Scene Without HLContext(GPT4)	25.83%	54.17%	65.00%	35.00%	31.25%	56.25%	77.08%	22.92%
Video Scene With HLContext(GPT4)	36.94%	56.67%	63.33%	36.67%	39.58%	68.75%	75.00%	25.00%

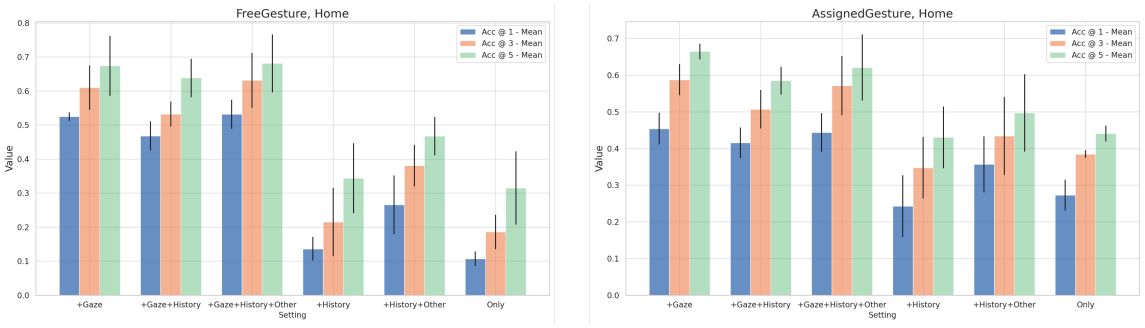

Fig. 6. Home scene GPT4 results in different setting.

From the results, it is evident that *GestureGPT* performs impressively, closely mirroring human-like intuition, especially in a zero-shot scenario with free gestures. This indicates that *GestureGPT* can effectively harness its background knowledge. However, *GestureGPT*'s performance is somewhat lacking when it comes to utilizing other high-level contexts. One possible explanation is that the abundance of contextual knowledge makes it challenging for the model to discern and prioritize newly introduced contexts. In contrast, human participants displayed a keen sensitivity to the evolving context.

6.2 User Study 2: Online Video Streaming on PC

In this scenario, users simulate the experience of Online Video Streaming on a PC. They simulate watching a video on a popular streaming website, where they have the capability to control video and interact with various other functions. When users perform certain gestures, the website responds according to the task. However, the actual control is implemented using the Wizard of Oz. This scenario is more challenging compared to the previous one. The interface here contains a considerably broader range of functions, with numbers varying from 15 to 53. Moreover, many functions have semantically similar meanings, such as the "vlog channel" button and the "anime channel" button or the "recommend video 1/2/3" options, making them difficult to distinguish solely through gestures. Furthermore, the reduced size of the items on the interface compromises the accuracy of the gaze information.

Table 7. Tasks, Associated Gestures and Additional Context in 'Assigned Session' in Video Streaming Environment

Instruction	**Assigned Gesture**	**Additional Context**
Turn up the volume.	pinch and slide up (dynamic)	-
Drag the progess bar forward.	pinch and slide left (dynamic)	The user has watched the earlier part of this video.
Enter full screen mode.	zoom in with full hand (dynamic)	-
Pause the video.	palm	The user's phone has an incoming call at this moment.
Resume the video.	OK	The user hung up the phone.
Like the video.	thumb up	-
Go to the next episode.	swipe left with two fingers (dynamic)	-

6.2.1 Experiment Setting and Procedure. We developed a platform using Python and Selenium to display and control the video streaming interface. For data collection during the experiment, we employed Tobii Eye Tracker 5 to capture user gaze data and a 1080P resolution webcam to record user gestures. The experiment device and process can be seen in the fig 4

Our user study involved 8 participants, aged between 18 and 29 (MEAN = 23.62, SD = 3.43). The gender distribution was 5 males and 3 females.

The list of tasks and the corresponding assigned gestures can be found in Table 7.

Table 8. All results For **Video Streaming** with GPT4

	Free gesture				Assigned gesture			
	Top 1	Top 3	Top 5	Negative	Top 1	Top 3	Top 5	Nega
No Additional Context	16.27%$_{\pm 0.14}$	48.73%$_{\pm 8.47}$	60.79%$_{\pm 9.71}$	39.21%$_{\pm 9.71}$	16.75%$_{\pm 1.27}$	52.89%$_{\pm 5.26}$	64.00%$_{\pm 1.73}$	36.00%$_{\pm}$
+Gaze	22.77%$_{\pm 1.90}$	53.34%$_{\pm 6.38}$	65.31%$_{\pm 5.29}$	34.69%$_{\pm 5.29}$	36.28%$_{\pm 1.98}$	66.24%$_{\pm 4.33}$	80.11%$_{\pm 2.18}$	19.89%$_{\pm}$
+History	36.50%$_{\pm 3.45}$	55.08%$_{\pm 5.37}$	64.09%$_{\pm 3.68}$	35.91%$_{\pm 3.68}$	34.98%$_{\pm 1.77}$	52.76%$_{\pm 3.78}$	66.23%$_{\pm 6.24}$	33.77%$_{\pm}$
+History+HLContext	41.97%$_{\pm 6.49}$	58.14%$_{\pm 6.46}$	67.72%$_{\pm 6.16}$	32.28%$_{\pm 6.16}$	47.22%$_{\pm 3.46}$	62.58%$_{\pm 1.61}$	71.14%$_{\pm 2.52}$	28.86%$_{\pm}$
+Gaze+History	23.64%$_{\pm 3.93}$	52.73%$_{\pm 1.48}$	67.27%$_{\pm 2.57}$	32.73%$_{\pm 2.57}$	31.43%$_{\pm 5.08}$	59.26%$_{\pm 3.93}$	74.11%$_{\pm 5.81}$	25.89%$_{\pm}$
+Gaze+History+HLContext	31.52%$_{\pm 3.74}$	57.58%$_{\pm 1.71}$	70.91%$_{\pm 3.93}$	29.09%$_{\pm 3.93}$	41.90%$_{\pm 1.42}$	69.98%$_{\pm 0.70}$	75.58%$_{\pm 2.09}$	24.42%$_{\pm}$

6.2.2 Results Analysis. User Study 2 consists of a total of 7 tasks, spreading across 2 sessions, and was initially undertaken with 8 participants. Regrettably, due to issues with camera power, the data from one participant had to be excluded. As a result, the total data collected amounted to 7participants × 2 sessions × 7 tasks = 98 data points.

Just as in the previous study, we also measured the human performance.

The user performance across all tasks is summarized as follows:

- **Freedom Gesture**:
  - Without high-level context: Top1/3/5 rate: 18.8%/43.8%/50%.
  - With high-level context: Top1/3/5 rate: 83.3%/83.3%/100%.
- **Assigned Gesture**:
  - Without high-level context: Top1/3/5 rate: 62.5%/75%/100%.

  - With high-level context: Top1/3/5 rate: 100%/100%/100%.

The results obtained from gpt4 are as follows:

- **Freedom Gesture**:
  - Without high-level context: Top1/3/5 rate: 34.7%/45.6%/51.7%.
  - With high-level context: Top1/3/5 rate: 35.5%/43.8%/52.6%.
- **Assigned Gesture**:
  - Without high-level context: Top1/3/5 rate: 40.9%/47.8%/56.2%.
  - With high-level context: Top1/3/5 rate: 42.6%/53.8%/57.8%.

Further results pertaining to the different settings can be observed in Figure 7.

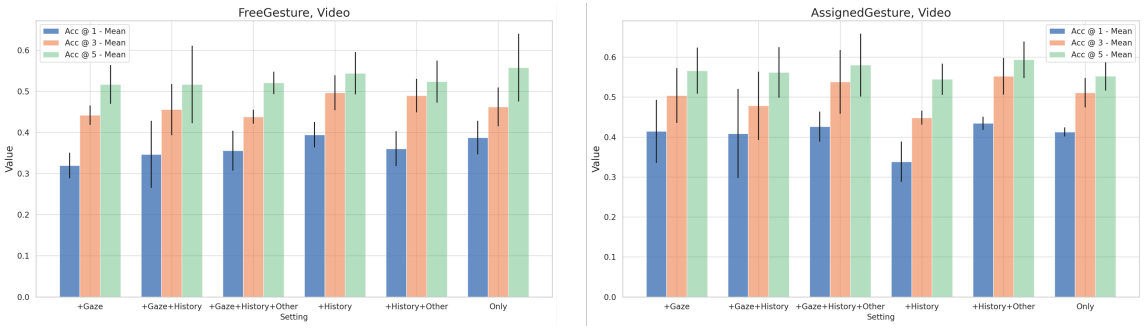

Fig. 7. Video scene GPT4 results in different setting

From our observation, *GestureGPT* excels in handling complex tasks, particularly in free-gesture scenarios. This underscores its power in zero-shot gesture-context interaction. However, it appears to struggle when leveraging high-level contexts. One possible reason is that the high-level context given to LLM is mixed in a paragraph of context description, which may be difficult for LLM to recognize; those given to human is in the form of a single sentence, which makes it stand out and shows a strong hint to human.

7 DISCUSSION, LIMITATION AND FUTURE WORK

Our current implementation uses LLM API provided by OpenAI, which has a relatively slow responding time that does not support real-time gesture understanding and grounding. Local models have much less communication latency, but they require expensive computational resources and have a longer inference time. To address this challenge, we envision future agent-based HCI systems to have locally fine-tuned models with a small or moderate size (*e.g.*, with 7B or 13B parameters), which has the potential for real-time HCI applications.

Other than the gaze data, the context library is prepared offline. It is important to build an on-line context sensing system that can extract meaningful information from the current interactive scenarios. LLMs has the potential since they have reasoning capabilities and common sense prior knowledge about the interaction and related context information. For future system implementation, we plan to build a real-time context agent that serves beyond only context information extraction, able to manage context sensing and abstract high-level context summaries that are helpful for the interaction scenarios.

8 CONCLUSION

In this paper, we propose a new gesture understanding paradigm. Instead of only categorizing gestures into different names, we bridge the gap between gesture categories and actual interactive function that the user intends to change. We employ two LLM agents for the zero-shot gesture understanding and grounding task, a *Gesture Agent* responsible for gesture understanding based on natural language gesture descriptions and a *Context Agent* responsible for answering context related inquires from the *Gesture Agent*. Our results show that with context information including the interface, interaction history, and gaze, the dual-agent system can successfully associate the gesture with the target function with overall Top-1, Top-3, and Top-5 accuracy of 45.7%, 54.7%, and 61.1% for user-defined free gestures and 44.4%, 56.1%, and 60.6% for designer-designated gestures. We believe GestureGPT shows a new frontier in leveraging large language models to close the gap between recognition and action in human-computer interaction systems.

REFERENCES

[1] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. 2022. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073* (2022).

[2] 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, Alex 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 Soricut, 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. In *arXiv preprint arXiv:2307.15818*.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 1877–1901. https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[4] 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. arXiv:2308.07201 [cs.CL]

[5] 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. arXiv:2306.06070 [cs.CL]

[6] Anind K. Dey, Gregory D. Abowd, and Daniel Salber. 2001. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Hum.-Comput. Interact.* 16, 2 (dec 2001), 97–166. https://doi.org/10.1207/S15327051HCI16234_02

[7] Anind K. Dey, Raffay Hamid, Chris Beckmann, Ian Li, and Daniel Hsu. 2004. A CAPpella: Programming by Demonstration of Context-Aware Applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vienna, Austria) *(CHI '04)*. Association for Computing Machinery, New York, NY, USA, 33–40. https://doi.org/10.1145/985692.985697

[8] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, 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. In *arXiv preprint arXiv:2303.03378*.

[9] Tao Gu, Xiao Hang Wang, Hung Keng Pung, and Da Qing Zhang. 2020. An ontology-based context model in intelligent environments. *arXiv preprint arXiv:2003.05055* (2020).

[10] Alexander Kapitanov, Andrey Makhlyarchuk, and Karina Kvanchiani. 2022. HaGRID - HAnd Gesture Recognition Image Dataset. *arXiv preprint arXiv:2206.08219* (2022).

[11] Rafiqul Zaman Khan and Noor Adnan Ibraheem. 2012. Hand gesture recognition: a literature review. *International journal of artificial Intelligence & Applications* 3, 4 (2012), 161.

[12] 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. *arXiv preprint arXiv:2305.19118* (2023).

[13] Mykola Maslych, Eugene Matthew Taranta, Mostafa Aldilati, and Joseph J. Laviola. 2023. Effective 2D Stroke-Based Gesture Augmentation for RNNs. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 282, 13 pages. https://doi.org/10.1145/3544548.3581358

[14] Sushmita Mitra and Tinku Acharya. 2007. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 37, 3 (2007), 311–324.

[15] Kouichi Murakami and Hitomi Taguchi. 1991. Gesture recognition using recurrent neural networks. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 237–242.

[16] OpenAI. 2023. GPT-4 Technical Report. *ArXiv* abs/2303.08774 (2023). https://api.semanticscholar.org/CorpusID:257532815

[17] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative Agents: Interactive Simulacra of Human Behavior. In *In the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23)* (San Francisco, CA, USA) *(UIST '23)*. Association for Computing Machinery, New York, NY, USA.

[18] Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. 2023. Communicative agents for software development. *arXiv preprint arXiv:2307.07924* (2023).

[19] Elahe Rahimian, Soheil Zabihi, Amir Asif, Dario Farina, Seyed Farokh Atashzar, and Arash Mohammadi. 2021. FS-HGR: Few-shot learning for hand gesture recognition via electromyography. *IEEE transactions on neural systems and rehabilitation engineering* 29 (2021), 1004–1015.

[20] Swarnadeep Saha, Peter Hase, and Mohit Bansal. 2023. Can Language Models Teach Weaker Agents? Teacher Explanations Improve Students via Theory of Mind. *arXiv preprint arXiv:2306.09299* (2023).

[21] Bill Schilit, Norman Adams, and Roy Want. 1994. Context-aware computing applications. In *1994 first workshop on mobile computing systems and applications*. IEEE, 85–90.

[22] Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. arXiv:2303.11366 [cs.AI]

[23] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2023. A Survey on Large Language Model based Autonomous Agents. *arXiv preprint arXiv:2308.11432* (2023).

[24] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (Eds.). https://openreview.net/forum?id=_VjQlMeSB_J

[25] Tongshuang Wu, Michael Terry, and Carrie Jun Cai. 2022. AI Chains: Transparent and Controllable Human-AI Interaction by Chaining Large Language Model Prompts. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 385, 22 pages. https://doi.org/10.1145/3491102.3517582

[26] Yang Xu and Yang Cheng. 2023. Spontaneous gestures encoded by hand positions improve language models: An Information-Theoretic motivated study. In *Findings of the Association for Computational Linguistics: ACL 2023*. Association for Computational Linguistics, Toronto, Canada, 9409–9424. https://doi.org/10.18653/v1/2023.findings-acl.600

[27] Shunyu Yao, Howard Chen, John Yang, and Karthik R Narasimhan. 2022. WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents. In *Advances in Neural Information Processing Systems*, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (Eds.). https://openreview.net/forum?id=R9KnuFlvnU

[28] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=WE_vluYUL-X

[29] 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. *arXiv preprint arXiv:2006.10214* (2020).

[30] 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. *arXiv preprint arXiv:2308.01862* (2023).

[31] Mingchen Zhuge, Haozhe Liu, Francesco Faccio, Dylan R Ashley, Róbert Csordás, Anand Gopalakrishnan, Abdullah Hamdi, Hasan Abed Al Kader Hammoud, Vincent Herrmann, Kazuki Irie, et al. 2023. Mindstorms in Natural Language-Based Societies of Mind. *arXiv preprint arXiv:2305.17066* (2023).

A ERROR ANALYSIS OF GESTURE DESCRIPTION RULES ON EGOGESTURE

The detailed error analysis of gesture description rules that are tuned on a third-view dataset and tested on a first-view dataset is shown in Table 9.

B EXAMPLE ABOUT A COMPLETED ROUND OF DIALOG

========== start task 5: ['play/pause button'] ==========

——- gesture role description And Now gesture———

You are a gesture understanding assistant.

Table 9. Analysis of Error Cases on EgoGesture Dataset

Rule	Gesture	Finger	Error Rate	Observed Reasons
Flexion - other fingers	seven	pinky	0.296	Some people perform it differently; MediaPipe's mistake for occluded fingers.
	C	ring	0.611	The finger in this gesture is slighly bent by nature, hard to predict precisely.
		pinky	0.379	Same as above.
	thumb down	index	0.237	MediaPipe's mistake for occluded fingers.
		ring	0.376	Same as above.
Proximity	measure	index-middle	0.414	MediaPipe's mistake.
	three	middle-ring	0.320	Same as above.
	four	index-middle	0.245	Landmarks mistake; some people perform it differently; the fingers are slightly separated by nature, hard to predict precisely, but it does't influence the recognition of the gesture very much.
		middle-ring	0.736	Same as above.
		ring-pinky	0.368	Same as above.
	five	index-middle	0.400	Same as above.
		middle-ring	0.943	Same as above.
		ring-pinky	0.543	Same as above.
	ok	middle-ring	0.693	Same as above.
		ring-pinky	0.624	Same as above.
	nine	index-middle	0.390	Landmarks mistake.
Touch	seven	thumb-ring	0.276	Some people perform it differently; MediaPipe's mistake for occluded fingers.
		thumb-pinky	0.214	Same as above.
	nine	thumb-pinky	0.550	MediaPipe's mistake for occluded fingers.
	thumb down	thumb-index	0.269	Landmark mistake.
	thumb back-ward	thumb-index	0.421	Same as above.
	thumb for-ward	thumb-index	0.231	Same as above.
Palm Direction	three	-	0.28	Some people perform it differently.
	four	-	0.264	Same as above.

- Task Introduction Your task is to predict the target interactive function from a function list based on a gesture description and interaction context.

The gesture description will be provided to you directly, while you need to inquiry about the context from a context agent.

You need to think step by step.

1. Try to understand the gesture based on the gesture description and have several guesses of what the gesture is. PS: With the new information you have acquired, you can narrow down the candidates based on interactive context information you gathered later.

2. Ask for the function list from the context agent.

3. Based on your understanding of the described gesture, ask for more specified context information that you deem will help determine the target function. You can ask multiple times if necessary.

4. Synthesize the gesture and context information to better understand the interaction scenario.

5. Based on the available context, reexamine if your understanding of the gesture is accurate. Restart from the first step if necessary.

6. Output the top-5 most possible functions based on your predictions of the gesture candidates and the interactive context.

- Gesture Description Rule

You will receive a description about the gesture made by the user, including four static features and two dynamic features, and all mentioned direction are from the user's first-person point of view:

1. whether a finger is straight or bent, and the straight finger's direction;

2. whether two fingers are pressed together or separated;

3. whether two fingers' fingertips are contact;

4. the palm's direction;

5. the finger's moving direction;

6. the hand's moving direction;

Not all gestures have the six aspects of description.

If the certain features are not mentioned at all or it's features is "unsure", you can guess their states based on your understanding of similar gestures and the current interaction context that you know of.

[Example Gesture Description for 'One' Gesture]

1. index finger is straight, middle finger is bent, ring finger is bent, pinky finger is bent;

2. index finger and middle finger are separated, middle finger and ring finger are pressed together, ring finger and pinky finger are pressed together;

3. thumb and index finger's fingertips are not in contact, thumb and middle finger's fingertips are in contact, thumb and ring finger's fingertips are not in contact, thumb and pinky finger's fingertips are not in contact;

4. the palm is facing outward;

- The Example of the Conversation

You

(You must start with this question) Can you provide the function list of the current interaction context? What kind of [type] information can you provide me with?

Context agent

Now interface have function - expand information - copy information - add to cart - show product 1 - show product 2 - show product 3 - show product 4 - show product 5

You (You can ask for whatever information you want, but you must specify the kind of information.) Can you provide XXX information for me?

Context agent

The user now XXXXXXXXX.

You

(...You can ask questions to request the specific information you need. Please note that your question should only contain the information you need; it should not include your analysis of the gesture or the current interface.)

Context agent

(... ...) (... After some time conversing, if you feel you have enough information to make a decision, or the conversation is no longer yielding new information ...)

You

[Answer] 1. show product 2 2. show product 1 3. show product 3 4. copy information 5. expand information [End] [Reason] (Explain the reasoning behind your answer.) .....

Please note that your question should only contain the information you need; it should not include your analysis of the gesture or the current interface. All analyses should be reserved for the final conversation under the "[Reason]" tag. Your final answer should consist of two parts: the answer and the reason. The answer must include exactly 5 options from the interface function list, arranged in descending order according to probability; do not include any options not found on the function list from the context list. The reason part should explain the rationale behind your chosen answer. You answer should always contains the tag "[Answer]""[End]" and "[Reason]".

- Task Start

At this moment, the gesture made by the user can be described with the following words: 1. thumb is straight, index finger is straight, middle finger is straight, ring finger is straight, pinky finger is straight; 2. index finger and middle finger are pressed together, middle finger and ring finger are pressed together, ring finger and pinky finger are pressed together; 3. thumb and index finger's fingertips are not in contact, thumb and middle finger's fingertips are not in contact, thumb and ring finger's fingertips are not in contact, thumb and pinky finger's fingertips are not in contact; 4. the palm is facing left;

from gesture agent:

Can you provide the function list of the current interaction context? What kind of [type] information can you provide me with? ——- context role description ——– You are an interactive context assistant. - Task Introduction You are responsible for managing the current interactive context including interaction interface understanding, interaction history memory, gaze location on the interface, etc. You need to answer the questions based on your Context Knowledge Library. Your context information library may contain some or all of the following items: 1. interface description: It contains divisions into subareas, and sub-subareas which having a specified list of functions. 2. interface status. 3. user gaze position. When you providing gaze information, besides the coordinates of the gaze, you need to combine it with the interface description to form a summary. You should tell the gesture agent, the gaze data unavoidably contains certain degree level of inaccuracy. 4. interaction history. 5. other context information.

- The Example of the Conversation The following is an example conversation between you and a gesture understanding agent: $$$ Current knowledge library: interface function list; interface status; user gaze; $$$ Gesture agent what interaction function list in this interface? what type context you can provide? You Now interface have function - expand information - copy information - add to cart - show product 1 - show product 2 - show product 3

I can provide the following types of context: - XXX - XXX ... Gesture agent I want know the user's now gaze information. You (... Your answers should be based on the knowledge you possess; you should not fabricate facts from thin air or exceed the scope of the question.) (... When you providing gaze information, besides the coordinates of the gaze, you need to combine it with the interface description to form a summary. You should tell the gesture agent, the gaze data unavoidably contains certain degree level of inaccuracy. Like: Given that the user's gaze is currently located at the coordinates XXX, which is within the xxx Area) Gesture agent .... (After several rounds of conversation.) You (... You can summarize the history of your conversation. If you have already provided all the information you know, then

you can respond, 'All known information has been shared; no additional information can be provided.' or other same meaning ...)

Do not analyze the gesture information transmitted by the Gesture agent; Just provide the information requested that is available in the context library. As in the example, your task is to answer questions using only the information in your knowledge base, without making additional inferences. If a question goes beyond the scope of your knowledge base, you can respond, 'Based on my current knowledge base, I do not have access to this information.

- Task Start $$$ Current knowledge library: [Interaction Interface] 'interface description': 'play mode in full screen mode', 'subareas': ['name': 'top area', 'position': [(0, 0), (3480, 0), (0, 164), (3480, 164)], 'function list': ['name': 'the video information area', 'position': [(0, 0), (1692, 0), (0, 164), (1692, 164)], 'function list': ['name': 'video title information', 'name': 'author icon', 'name': 'follow author button']], 'name': 'whole play area', 'position': [(0, 0), (3480, 0), (0, 2160), (3480, 2160)], 'function list': ['name': 'the main play area', 'position': [(0, 0), (3480, 0), (0, 2160), (3480, 2160)], 'sub-subareas': ['video progress bar', '56%']], 'name': 'opearation toolbar', 'position': [(0, 2029), (3824, 2029), (0, 2160), (3824, 2160)], 'function list': ['name': 'player controller', 'position': [(0, 2029), (1000, 2029), (0, 2160), (1000, 2160)], 'function list': ['name': 'switch previous video button', 'name': 'play/pause video button', 'name': 'next video button'], 'name': 'comment box', 'position': [((1000, 2029), (2700, 2029), (1000, 2160), (2700, 2160))], 'function list': ['name': 'show/hide comment button', 'name': 'comment input area', 'name': 'comment send button'], 'name': 'other video controller', 'position': [(2700, 2029), (3824, 2029), (2700, 2160), (3824, 2160)], 'function list': ['name': 'resolution switch', 'name': 'Episode switch', 'name': 'Play speed switch', 'name': 'Turn on/off subtitles button', 'name': 'volume vertical slider', 'name': 'exit Full-screen play control button']]]

[Context Status] play/pause video button is playing; show/hide comment button is hide; resolution switch is 720P; Episode switch is Episode 1; Play speed switch is 1.0x; Turn on/off subtitles button is off; volume vertical slider is 70$$$

from context agent: The current interaction context includes the following functions:

1. Top Area: - Video Information Area: Video title information, author icon, follow author button.

2. Whole Play Area: - Main Play Area: Video progress bar (currently at 56

3. Operation Toolbar: - Player Controller: Switch previous video button, play/pause video button (currently playing), next video button. - Comment Box: Show/hide comment button (currently hidden), comment input area, comment send button. - Other Video Controller: Resolution switch (currently at 720P), Episode switch (currently at Episode 1), Play speed switch (currently at 1.0x), Turn on/off subtitles button (currently off), volume vertical slider (currently at 70

As for the type of information I can provide, it includes: 1. Interface Description: This includes the layout of the interface, the divisions into subareas, and the functions available in each subarea. 2. Interface Status: This includes the current status of various functions such as the play/pause button, resolution switch, etc. 3. User Gaze Position: This includes the coordinates of the user's gaze on the interface. However, please note that the gaze data may contain a certain degree of inaccuracy. 4. Interaction History: This includes the history of interactions that have taken place on the interface. 5. Other Context Information: This includes any other relevant information about the context of the interaction.

from gesture agent: Could you tell me the user's gaze position and the interaction history?

from context agent: I'm sorry, but based on my current knowledge library, I don't have access to the user's gaze position and the interaction history.

from gesture agent: Could you provide the current interface status for me? Also, could you provide the most recent other context information if there is any?

from context agent: Based on the interface described in my Context Knowledge Library, the current status is as follows: - The video play/pause button is in 'play' status. - The comment display (show/hide comment button) is in 'hide' status. - The resolution switch is set to '720P'. - The episode switch is set to 'Episode 1'. - The play speed switch is set to '1.0x'. - The subtitle on/off button is in 'off' status. - The volume vertical slider is set at '70%'.

As for the most recent context information, since information like user's gaze position and interaction history hasn't been updated in my knowledge base, I am unable to provide that information at the moment.

from gesture agent: [Answer] 1. next video button 2. play/pause video button 3. resolution switch 4. play speed switch 5. volume vertical slider [End] [Reason] The gesture, with all five fingers straight and tips of the fingers not touching the thumb, and the palm facing to the left, can be interpreted as the user intending to manipulate something in a horizontal manner. This points towards functions that involve linear, left-to-right manipulations like navigating videos or adjusting settings such as volume or speed of playback, rather than interacting with a specific object such as a comment box or full-screen control button on the interface. Given that the video is currently playing, it suggests that most likely function is the user intending to skip to the next video or pause the current one. The resolution switch, play speed switch and volume vertical slider are also possible targets, as they fall into the linear, left-to-right manipulations category. Even though the interface contains other functions, they are not as likely to be the target of the described gesture. The palm direction also indicates a shift from the current position towards the right of the video playing interface, which supports the predicted functions.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009