# Supplementary Material of "SIAgent: Spatial Interaction Agent via LLM-powered Eye-Hand Motion Intent Understanding in VR"

Zhimin Wang, Chenyu Gu, and Feng Lu, *Senior Member, IEEE*

## I. Additional Study: Impact of Latency on User Experience

To address the lack of subjective evaluation regarding latency in the Study 1, we conducted a supplementary study during the revision phase. We recruited three participants (all male, aged 25-30) from the pool of participants in Study 1. We employed a within-subjects design to compare two interaction methods: Gaze+Pinch (GP) and SIAgent (Agent, Ours). Participants were required to perform 18 trials (6 tasks $\times$ 3 scenes) described in the original manuscript using each interaction technique. Upon completion, participants filled out a post-study questionnaire regarding the impact of latency on the Agent interaction. The questionnaire consisted of the following four questions:

**Q1.** For the Agent interaction, during the waiting period immediately following your interaction, did you perceive the system as "processing" (thinking) or "hanging"? To what extent did this latency cause anxiety or disrupt your operational flow?

**Q2.** Consider the following trade-off: one interaction modality provides low latency but requires memorizing specific commands, while another involves higher latency but allows for natural and flexible expression. Which would you prefer, and could you explain why?

**Q3.** If dynamic visual feedback were provided during the wait (e.g., a progress bar or the virtual hand initiating movement early), would it mitigate your perception of the waiting time? Do you have any other suggestions for dynamic feedback mechanisms?

**Q4.** Reflecting on your previous responses, please rate your subjective experience of the system's latency on a scale of 1 to 10, where 1 signifies "completely intolerable" and 10 signifies "fully acceptable for continued use." Please provide a brief justification for your score.

**Subjective Analysis of Latency Perception.** We analyzed qualitative participant feedback to understand perceptions of latency, trade-offs regarding acceptance, and preferences for feedback mechanisms. The detailed analysis for each question is presented below:

*Regarding Q1 (Perception).* All participants (P1-P3) perceived the system state as "thinking" rather than "hanging" (unresponsive) during standard latency periods. P1 reported a sense of "calm" rather than anxiety, focusing on interface cues indicating that "intent is being analyzed." P2 rationalized the latency as necessary computation to accommodate individual variations in gesture habits. However, P3 noted that while standard latency is acceptable, sporadic outliers (significantly longer delays) could trigger anxiety regarding potential system crashes, highlighting the criticality of consistency.

*Regarding Q2 (Preference).* Participants demonstrated a strong preference for the Agent (natural interaction) over the low-latency manual method. P1 and P2 emphasized physical relief, noting that manual gestures induced higher fatigue, whereas the Agent interaction was described as "effortless." P3 focused on cognitive load, arguing that the burden of memorizing commands for complex interaction sets "rises linearly," demonstrating natural expression superior despite the associated latency.

*Regarding Q3 (Feedback Mechanisms).* Participants distinguished between opaque waiting states and transparent process visualization. P2 specifically noted that simple status indicators (e.g., a three-dot loading animation) could increase user anxiety due to the lack of visible progress. Instead, participants advocated for process visualization. P2 suggested sequentially highlighting detected target objects to demonstrate that the agent is "analyzing intent step-by-step," while P3 recommended explicitly visualizing the reasoning process (similar to streaming text in web-based LLMs). P1 added that a progress bar would facilitate proactive motor planning.

*Regarding Q4 (Acceptance).* Notably, all participants rated the experience an 8 out of 10. Acceptance was driven by the "intelligence payoff"; P1 described the accurate inference as "surprising" and 'magical," comparable to interactions with advanced LLMs. P2 valued the zero-learning curve compared to the requirement of learning Gaze+Pinch interactions in advance. Interestingly, P3 framed the processing time positively as a "rest period." P2 concluded that reducing the "black box" nature of the agent would further enhance this score.

*Conclusion.* The feedback demonstrates that latency is well-tolerated when it yields a tangible return in the form of reduced physical fatigue and cognitive effort. The primary design implication is to shift from static waiting states to informative visualization to maintain user trust.

Zhimin Wang, Chenyu Gu, and Feng Lu are with the State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing 100191, China. e-mail: {*zm.wang* \ *gucy* \ *lufeng*}@buaa.edu.cn.

Feng Lu is the corresponding author.

Manuscript received July 17, 2025.

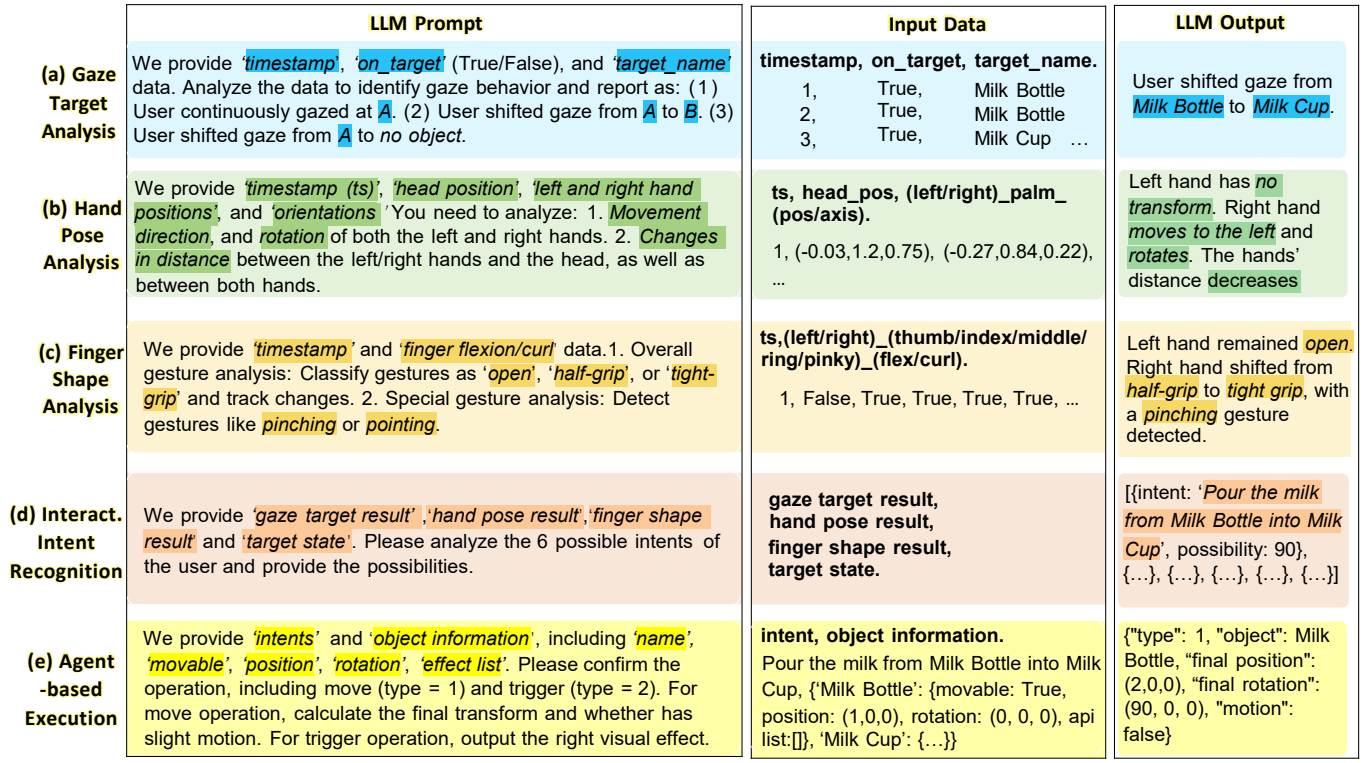


Fig. 1: Examples of LLM prompts, input data, and output responses used in our three-stage pipeline. (a) Gaze target analysis: translates eye-tracking data into natural language descriptions of user attention patterns. (b) Hand pose analysis: converts hand position and rotation data into linguistic descriptions of movement patterns. (c) Finger shape analysis: analyzes finger flexion and curl patterns to identify grasp types and gesture transitions. (d) Intent recognition: infers user interaction intents from combined gaze, hand pose, and finger shape descriptions, providing ranked intent predictions. (e) Agent-based execution: generates spatial operation parameters from recognized intents to drive virtual agent actions. The pipeline processes multimodal data in parallel during stages (a)-(c), combines results for intent inference in stage (d), and produces executable agent commands in stage (e). For more detailed prompt templates, please refer to Section III.

## II. Discussion on Handling Intent Misinterpretation

We address the handling of misinterpretation in two contexts: our experimental evaluation and practical application.

In our experimental setup (Study 1), we adopted a decoupled strategy to evaluate Intent Recognition and Agent Execution separately. Specifically, for the Agent Execution evaluation, we controlled the input by providing ground truth intents (simulating 100% recognition accuracy). This design was intentionally chosen to isolate the Agent's execution performance from potential intent recognition errors, ensuring that the measured success rates reflect the Agent's planning and control capabilities rather than upstream interpretation failures.

Regarding practical application, we propose implementing specific fallback strategies in future deployments. If the system were to misinterpret an intent, a conversational repair mechanism could prompt the user to rephrase their expression. Alternatively, the user can revert to the standard Gaze+Pinch manipulation technique if intent recognition fails.

TABLE I: Task Definitions, Visual Outcomes, Success Metrics and Prerequisite Task ID

| ID | Task Name | Task Type | Expected Visual Output (User Perspective) | Success Metric (System Logic) | Pre. Task ID |
|---|---|---|---|---|---|
| | | | **StudyRoom** | | |
| 1 | Open **Study Door** | Trigger | **Study Door** rotates to open position | Trigger **Study Door**'s **Open** API | / |
| 2 | Open **Book** | Trigger | **Book** cover is flipped open | Trigger **Book**'s **Open** API | / |
| 3 | Write on **Book** with **Red Pencil** | Movement | **Red Pencil** traces appear on **Book** | **Red Pencil** IN **Book**'s bounding box | 2 |
| 4 | Put **Red Pencil** in **Pen Holder** | Movement | **Red Pencil** rests inside **Pen Holder** | **Red Pencil** IN **Pen Holder**'s bounding box | 3 |
| 5 | Open **Laptop** | Trigger | **Laptop** screen lid lifts up | Trigger **Laptop**'s **Open** API | / |
| 6 | Turn On **Laptop** | Trigger | **Laptop** screen illuminates | Trigger **Laptop**'s **Turn On** API | 5 |
| 7 | Open **Drawer** | Trigger | **Drawer** slides out | Trigger **Drawer**'s **Open** API | / |
| 8 | Put **Purple Pencil** in **Drawer** | Movement | **Purple Pencil** rests inside **Drawer** | **Purple Pencil** IN **Drawer**'s bounding box | 7 |
| 9 | Turn On **Desk Lamp** | Trigger | **Desk Lamp** emits light | Trigger **Desk Lamp**'s **Turn On** API | / |
| 10 | Adjust **Desk Lamp** | Trigger | **Desk Lamp** brightness intensity increases | Trigger **Desk Lamp**'s **Adjust** API | 9 |
| 11 | Write on **Sticky Note** with **Pink Marker** | Movement | **Pink Marker** traces appear on **Sticky Note** | **Pink Marker** IN **Sticky Note**'s bounding box | / |
| 12 | Pull Open **Curtain** | Trigger | **Curtain** retracts upwards | Trigger **Curtain**'s **Open** API | / |
| 13 | Open **Window** | Trigger | **Window** pane slides open | Trigger **Window**'s **Open** API | 12 |
| 14 | Close **Window** | Trigger | **Window** pane closes completely | Trigger **Window**'s **Close** API | 13 |
| 15 | Write on **Whiteboard** with **Yellow Marker** | Movement | **Yellow Marker** traces appear on **Whiteboard** | **Yellow Marker** IN **Whiteboard**'s bounding box | / |
| 16 | Clean **Whiteboard** with **Eraser** | Movement | Traces disappear from **Whiteboard** | **Eraser** IN **Whiteboard**'s bounding box | 15 |
| 17 | Open **Wardrobe** | Trigger | **Wardrobe** doors swing open | Trigger **Wardrobe**'s **Open** API | / |
| 18 | Put **Box** in **Wardrobe** | Movement | **Box** rests on **Wardrobe** shelf | **Box** IN **Wardrobe**'s bounding box | 17 |
| 19 | Play **Guitar** | Trigger | **Guitar** emits sound | Trigger **Guitar**'s **Play** API | / |
| | | | **BedRoom** | | |
| 20 | Turn Off **Bedside Lamp** | Trigger | **Bedside Lamp** goes dark | Trigger **Bedside Lamp**'s **Turn Off** API | / |
| 21 | Open **Nightstand Drawer** | Trigger | **Nightstand Drawer** slides out | Trigger **Nightstand Drawer**'s **Open** API | / |
| 22 | Put **Book** in **Nightstand Drawer** | Movement | **Book** rests inside **Nightstand Drawer** | **Book** IN **Nightstand Drawer**'s bounding box | 21 |
| 23 | Open **Washing Machine Lid** | Trigger | **Washing Machine Lid** lifts up | Trigger **Washing Machine**'s **Open Lid** API | / |
| 24 | Put **Clothes** in **Washing Machine** | Movement | **Clothes** are inside **Washing Machine** | **Clothes** IN **Washing Machine**'s bounding box | 23 |

Continued on next page

**TABLE I – Continued from previous page**

| ID | Task Name | Type | Expected Visual Output (User Perspective) | Success Metric (System Logic) | Pre Task ID |
|---|---|---|---|---|---|
| 25 | Start **Washing Machine** | Trigger | **Washing Machine** emits start signal | Trigger **Washing Machine**'s **Start** API | 24 |
| 26 | Put **Pants** in **Storage Box** | Movement | **Pants** rest inside **Storage Box** | **Pants** IN **Storage Box**'s bounding box | / |
| 27 | Turn On **TV** | Trigger | **TV** screen displays content | Trigger **TV**'s **Turn On** API | / |
| 28 | Change **TV** Channel | Trigger | **TV** screen content changes | Trigger **TV**'s **Change Channel** API | 27 |
| 29 | Clean **TV** with **Towel** | Movement | **Towel** wipes **TV** screen surface | **Towel** IN **TV**'s bounding box | / |
| 30 | Open **Milk Bottle** Lid | Trigger | **Milk Bottle** Lid separates from **Milk Bottle** | Trigger **Milk Bottle**'s **Open Lid** API | / |
| 31 | Pour **Milk** into **Cup** | Movement | Milk flows from **Milk Bottle** into **Cup** | **Milk Bottle** IN **Cup**'s bounding box | 30 |
| 32 | Move **Cup** to **Left Nightstand** | Movement | **Cup** rests on **Left Nightstand** | **Cup** IN **Left Nightstand**'s bounding box | 31 |
| 33 | Water **Plant** with **Watering Can** | Movement | Water flows from **Watering Can** onto **Plant** | **Watering Can** IN **Plant**'s bounding box | / |
| 34 | Turn On **Phone** | Trigger | **Phone** screen illuminates | Trigger **Phone**'s **Turn On** API | / |
| 35 | Charge **Phone** | Trigger | **Phone** charging icon appears | Trigger **Phone**'s **Charge** API | / |
| 36 | Close **Window** | Trigger | **Window** pane closes | Trigger **Window**'s **Close** API | / |
| 37 | Clean **Window** with **Towel** | Movement | **Towel** wipes **Window** glass surface | **Towel** IN **Window**'s bounding box | / |
| | **LivingRoom & Kitchen** | | | | |
| 38 | Close **TV Cabinet** | Trigger | **TV Cabinet** door closes | Trigger **TV Cabinet**'s **Close** API | / |
| 39 | Open **TV Cabinet** | Trigger | **TV Cabinet** door opens | Trigger **TV Cabinet**'s **Open** API | 38 |
| 40 | Put **CD** in **TV Cabinet** | Movement | **CD** rests inside **TV Cabinet** | **CD** IN **TV Cabinet**'s bounding box | 39 |
| 41 | Light **Candle** | Trigger | Flame appears on **Candle** wick | Trigger **Candle**'s **Light** API | / |
| 42 | Extinguish **Candle** | Trigger | **Candle** flame disappears | Trigger **Candle**'s **Extinguish** API | 41 |
| 43 | Throw **Candle** in **Trash Can** | Movement | **Candle** rests in **Trash Can** | **Candle** IN **Trash Can**'s bounding box | 42 |
| 44 | Empty **Trash Can** | Trigger | **Trash Can** contents disappear | Trigger **Trash Can**'s **Empty** API | 43 |
| 45 | Check Time on **Clock** | Trigger | UI notification shows **Clock** time | Trigger **Clock**'s **Check Time** API | / |
| 46 | Put **Bread** on **Plate** | Movement | **Bread** rests on **Plate** surface | **Bread** IN **Plate**'s bounding box | / |
| 47 | Spread **Butter** on **Bread** | Movement | **Butter** covers **Bread** | **Butter** IN **Bread**'s bounding box | 46 |
| 48 | Cut **Bread** with **Knife** | Movement | **Knife** inserts into **Bread** | **Knife** IN **Bread**'s bounding box | 47 |
| 49 | Put **Pan** on **Stove** | Movement | **Pan** rests on **Stove** burner | **Pan** IN **Stove**'s bounding box | / |
| 50 | Turn On **Range Hood** | Trigger | **Range Hood** emits start signal | Trigger **Range Hood**'s **Turn On** API | / |
| 51 | Turn Off **Range Hood** | Trigger | **Range Hood** emits stop signal | Trigger **Range Hood**'s **Turn Off** API | 50 |

Continued on next page

**TABLE I – Continued from previous page**

| ID | Task Name | Type | Expected Visual Output (User Perspective) | Success Metric (System Logic) | Pre Task ID |
|---|---|---|---|---|---|
| 52 | Turn On **Stove** | Trigger | **Stove** indicator light turns on | Trigger **Stove**'s **Turn On** API | / |
| 53 | Turn Off **Stove** | Trigger | **Stove** indicator light turns off | Trigger **Stove**'s **Turn Off** API | 52 |
| 54 | Put **Fried Fish** on **Pan** | Movement | **Fried Fish** rests inside **Pan** | **Fried Fish** IN **Pan**'s bounding box | 49 |
| 55 | Sprinkle **Seasoning** on **Fried Fish** | Movement | **Seasoning jar** is positioned above **Fried Fish** | **Seasoning jar** IN **Fried Fish**'s bounding box | 54 |
| 56 | Pour **Coffee** into **Cup** | Movement | Coffee flows from **Coffee Pot** into **Cup** | **Coffee Pot** IN **Cup**'s bounding box | / |
| 57 | Put **Cup** in **Sink** | Movement | **Cup** rests in **Sink** basin | **Cup** IN **Sink**'s bounding box | 56 |
| 58 | Turn On **Faucet** | Trigger | Water flows from **Faucet** | Trigger **Faucet**'s **Turn On** API | 57 |
| 59 | Open **Sliding Door** | Trigger | **Sliding Door** slides along track | Trigger **Sliding Door**'s **Open** API | / |
| 60 | Close **Sliding Door** | Trigger | **Sliding Door** closes completely | Trigger **Sliding Door**'s **Close** API | 59 |

## III. Detailed Prompt Templates

### A. Phase 1: Spatial-to-Linguistic Translation

**Prompt 1: Gaze Target Analysis**

**Role:** Intelligent Agent in a VR environment. **Task:** Analyze the user's gaze behavior based on the collected gaze data. **Input Data:** {{`gaze data`}}

- `timestamp`: Indicating data is collected sequentially; you need to process and analyze based on the time series.
- `on_target`: Whether the user is currently gazing at an interactive object. If `TRUE`, the user is gazing at an object; if `FALSE`, the gaze point is not on any interactive object.
- `target_name`: The name of the object being gazed at (meaningful only when `on_target=TRUE`).

**Behavior Categories:** The user's gaze object behavior can be divided into three categories: 1. Continuously gazing at a specific object. 2. Shifting gaze from one object to another. 3. Initially gazing at an object, then shifting to no interactive object (`on_target` becomes `FALSE`).

**Instructions:** In this step, you need to analyze the corresponding gaze behavior based on the provided time-series data and provide the analysis result. According to the classification above, the format of the analysis result should be: "User continuously gazes at XXX" OR "User shifts gaze from XXX to YYY" OR "User shifts gaze from XXX to not gazing at any object" (where XXX and YYY need to be replaced based on the actual `target_name` field).

**Two points to emphasize:** 1. Please perform analysis based on `timestamp`; do not ignore the temporal relationship between data points. 2. Due to uncertainty during collection, some noise points may appear (e.g., a single `on_target=FALSE` record appearing in a series of `on_target=TRUE` data, or a few `target_name=YYY` records appearing in a series of `target_name=XXX` data). You need to identify and discard these noise points, extracting only those gaze objects that appear frequently.

**Output:** Please output only one line of analysis result following the format requirements above. The analysis process is done internally by you and does not need to be output.

**Prompt 2: Hand Pose Analysis**

**Role:** Intelligent Agent in a VR environment. **Task:** Calculate the user's hand behavior based on the collected hand data. **Input Data:** {{`hand data`}}

- `timestamp`: Indicating data is collected sequentially; you need to process and analyze based on the time series.
- `left_hand_pos_x/y/z`: Left hand position coordinates.
- `right_hand_pos_x/y/z`: Right hand position coordinates.
- `left_hand_rot_angle`: Left hand rotation angle (current frame relative to previous frame).
- `right_hand_rot_angle`: Right hand rotation angle (current frame relative to previous frame).
- `left/rightHand_head_dist`: Distance between left/right hand and user's head.
- `leftHand_rightHand_dist`: Distance between left hand and right hand.

Coordinate direction definition: +x is Right, -x is Left, +y is Up, -y is Down, +z is Forward, -z is Backward.

**Calculate and analyze the following 3 items:**

**1. Movement Analysis** *Targets:* The 6 position fields (`pos_x/y/z` for both hands). *Method:* Analyze each field. If the difference between the average of the last few frames and the average of the first few frames exceeds 0.1, determine there is significant

movement in that direction. Example: (Avg of last 5 frames `left_hand_pos_x`) - (Avg of first 5 frames `left_hand_pos_x`). If difference $> 0.1$, left hand moves right explicitly; if $< -0.1$, moves left explicitly; otherwise, no explicit movement in left-right direction. Analyze other 5 fields similarly. Use directions: Forward, Backward, Left, Right, Up, Down (do not use axis names). *Result Format:* 1. Left hand has significant movement, direction is... (may have multiple) OR Left hand has no significant movement; 2. Right hand has significant movement, direction is... (may have multiple) OR Right hand has no significant movement.
**2. Rotation Analysis** *Targets:* `left/right_hand_rot_angle`. *Method:* If values in consecutive frames exceed or approach 10, determine specific hand has explicit rotation. Example: If 3 consecutive intermediate frames have `left_hand_rot_angle` $> 10$, left hand has explicit rotation. *Result Format:* 1. Whether left hand has explicit rotation; 2. Whether right hand has explicit rotation.
**3. Distance Analysis** *Targets:* The 3 distance fields. *Method:* Take avg of first few frames as Start Dist, middle few as Mid Dist, last few as End Dist. Use 0.08 as threshold to analyze change. Example: If Mid - Start $> 0.08$ and End - Mid $> 0.08$, distance continuously increases. If Mid - Start $> 0.08$ and End - Mid $< -0.08$, distance increases then decreases. *Result Format:* 1. Distance between left hand and head has explicit change, distance increases/decreases/increases then decreases/decreases then increases OR no explicit change; 2. (Same for right hand and head); 3. (Same for left hand and right hand).
**Output:** Total 7 items of analysis results, so strictly 7 lines. Please base analysis on `timestamp`, use full data (not just first frames), strictly calculate based on data but provide summary results (e.g., "distance increases", not values). Analysis process is internal.

**Prompt 3: Finger Shape Analysis**

**Role:** Intelligent Agent in a VR environment. **Task:** Calculate user gesture behaviors based on collected finger data. **Input Data:** {{`finger data`}}

- `timestamp`: Indicating data is collected sequentially; you need to process and analyze based on the time series.
- Fields ending in `flex`: Whether the finger joint connected to the palm is flexed (e.g., `left_thumb_flex`).
- Fields ending in `curl`: Whether the distal finger joint is bent (e.g., `right_thumb_curl`).
- `left/right_pinching`, `left/right_pointing`: Whether left/right hand has "pinching" or "pointing" gesture.

**Analyze in two aspects:**
**1. Overall Gesture Analysis** Ignore thumb data. Look at the other 4 fingers for each hand. Define 3 gestures per frame:

- **Open:** Other 4 fingers' flex and curl are FALSE.
- **Half-grip:** Other 4 fingers' flex is TRUE, curl is FALSE.
- **Tight-grip:** Other 4 fingers' curl is TRUE.

Select the most probable gesture allowing for error. Analyze temporal changes, removing noise (usually changes $\leq 2$ times). *Result Format:* "Continuously maintains XXX gesture" OR "Changes from XXX to YYY" OR "Changes from XXX to YYY then to ZZZ".
**2. Special Gesture Analysis** Analyze `pinching` and `pointing` fields. If a field is TRUE for more than 5 consecutive frames, the hand has that special gesture. Handle noise (only valid if ¿5 frames). *Result Format:* "Special gesture exists: XXX" OR "No special gesture".
**Final Output Format:** Left hand gesture change situation: [Left hand overall result]; Right hand gesture change situation: [Right hand overall result]; Left hand special gesture situation: [Left hand special result]; Right hand special gesture situation: [Right hand special result]. Output only in this format. Analysis process is internal.

## *B. Phase 2: Interaction Intent Recognition*

**Prompt 4: Interaction Intent Recognition**

**Role:** Intelligent Agent in a VR environment. **Task:** Analyze the user's most likely interaction intent based on provided gaze info, hand info, gesture info, and potential object state info. **Input:** Gaze info: {{`gaze result`}}, Hand info: {{`hand result`}}, Gesture info: {{`gesture result`}}, State info: {{`state info`}}. **Rules:** 1. If gaze info contains two interaction objects, first consider interaction intents involving both objects (consider order); then consider single object intents. 2. Initially, ignore provided object state info. Only when your analyzed intent is ambiguous (e.g., "open door" vs "close door" where direction is unclear), then add object state info to supplement analysis and determine intent. If the analyzed intent has no such ambiguity, you MUST ignore object state info to avoid interference.
**Output:** Analyze the 6 most likely interaction intents. Score each 0-100 (higher = more likely). Object names in intents must match gaze info. Output a list containing 6 dictionaries, each with "intention" and "possibility" fields.
**Format:**

```
[{"intention": Intent1, "possibility": Score1},
 {"intention": Intent2, "possibility": Score2},
 ...
 {"intention": Intent6, "possibility": Score6}]
```

Strictly follow this format. No analysis process needed.

### C. Phase 3: Agent-based Execution

**Prompt 5: Agent Operation Generation**

**Role:** Intelligent Agent in a VR environment. **Task:** Help user control scene objects based on interaction intent. **Input Data:** {{`intent data`}}

- `intention`: User's interaction intent (task to complete).
- `objects_info`: List of related objects (name, movable, position, rotation, size, api_list).

Coordinates: xz is horizontal plane, y is height.

**Analysis Steps:**

**1. Determine Unique Controlled Object** If intent has one object, control it. If two, use common sense to pick one to control. Always control exactly one object (`controlled_object`).

**2. Determine Operation Type**

- **Type 1:** Move/Rotate needed, object is movable, and api_list is irrelevant to intent. (Likely for 2-object intents).
- **Type 2:** api_list is not empty, and an API name is strongly related to intent (calling API completes intent). (Do not force connection; if no suitable API, set type=0).
- **Type 0:** Cannot be completed by above types.

**3. Calculate Specific Operation Info**

- If **type=0**: No operation.
- If **type=1**: Calculate `destination_position`. Check if small-range `motion` is needed after arrival (0=simple move/place; 1=writing/cutting/wiping). Check `incline` (1=pouring water/milk; 0=otherwise). If 2 objects, usually move controlled object to the other.
- If **type=2**: Analyze which specific API to use (must exist in provided api_list).

**4. JSON Output Format**

- Type0: {`"type":0`}
- Type1: {`"type":1,"controlled_object":"","destination_position":,"motion":,"incline":`}
- Type2: {`"type":2,"controlled_object":"","api":""`}

Output JSON only (start with {, end with }). No analysis text.