

HandProxy: Expanding the Affordances of Speech Interfaces in Immersive Environments with a Virtual Proxy Hand

Chen Liang
clumich@umich.edu
University of Michigan
Ann Arbor, MI, USA

Yuxuan Liu
liurick@umich.edu
University of Michigan
Ann Arbor, MI, USA

Martez Mott
mamott@microsoft.com
Microsoft Research
Redmond, WA, USA

Anhong Guo
anhong@umich.edu
University of Michigan
Ann Arbor, MI, USA

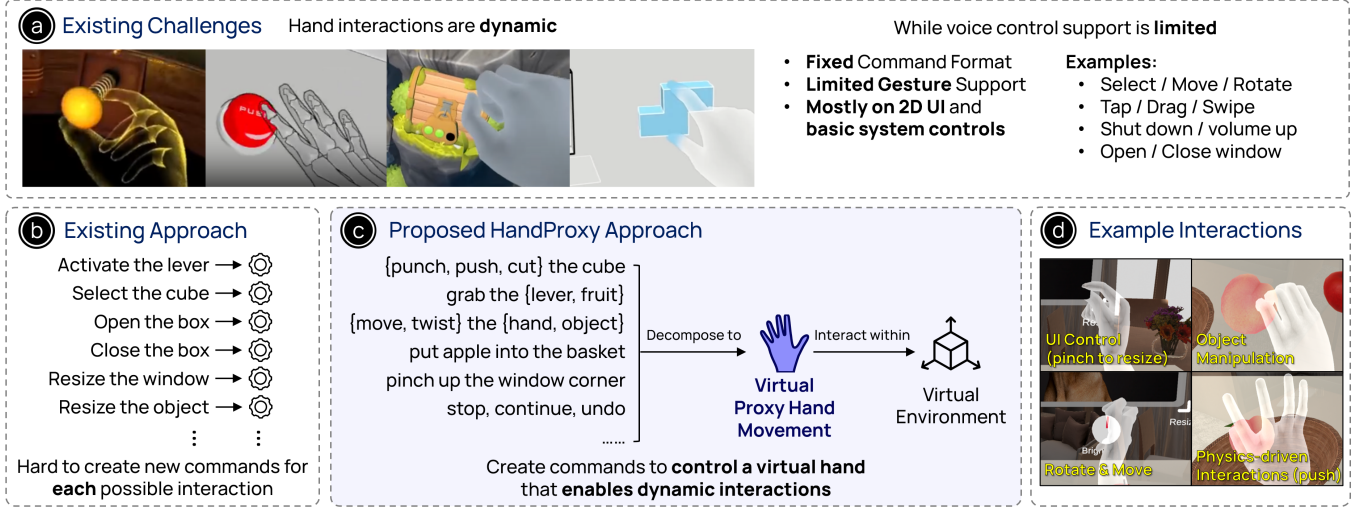


Figure 1: While XR devices increasingly support dynamic hand interactions, the corresponding speech-based interfaces remain limited (a). While adding speech commands enhances interface capability (b), they lack scalability for diverse hand interactions. We introduce the approach of using a virtual proxy hand that enables users to describe their desired interactions, which the system translates into executable hand movements (c). This approach supports various interactions, such as UI control, object manipulation, and interactions that emerge from physics, collisions, or hand rotation and movement (d).

Abstract

Hand interactions are increasingly used as the primary input modality in immersive environments, but they are not always feasible due to situational impairments, motor limitations, and environmental constraints. Speech interfaces have been explored as an alternative to hand input in research and commercial solutions, but are limited to initiating basic hand gestures and system controls. We introduce HandProxy, a system that expands the affordances of speech interfaces to support expressive hand interactions. Instead of relying on predefined speech commands directly mapped to possible interactions, HandProxy enables users to control the movement of a virtual hand as an interaction proxy, allowing them to describe the intended interactions naturally while the system

translates speech into a sequence of hand controls for real-time execution. A user study with 20 participants demonstrated that HandProxy effectively enabled diverse hand interactions in virtual environments, achieving a 100% task completion rate with an average of 1.09 attempts per speech command and 91.8% command execution accuracy, while supporting flexible, natural speech input with varying levels of control and granularity.

CCS Concepts

• **Human-centered computing** → **Interactive systems and tools.**

Keywords

Hand interaction, speech interface, interaction proxy, virtual environment, hand-object interaction

1 Introduction

Hand tracking is increasingly used as the primary input modality on extended reality (XR) devices to interact with the virtual environment [3, 4, 6]. As a result, many applications are built specifically for hand interactions, including games, productivity apps, and 2D/3D user interfaces (UI). For example, users can pinch and drag the



This work is licensed under a Creative Commons Attribution 4.0 International License.
IMWUT '25, September 2025
© 2025 Copyright held by the owner/author(s).
<https://doi.org/10.1145/3749484>

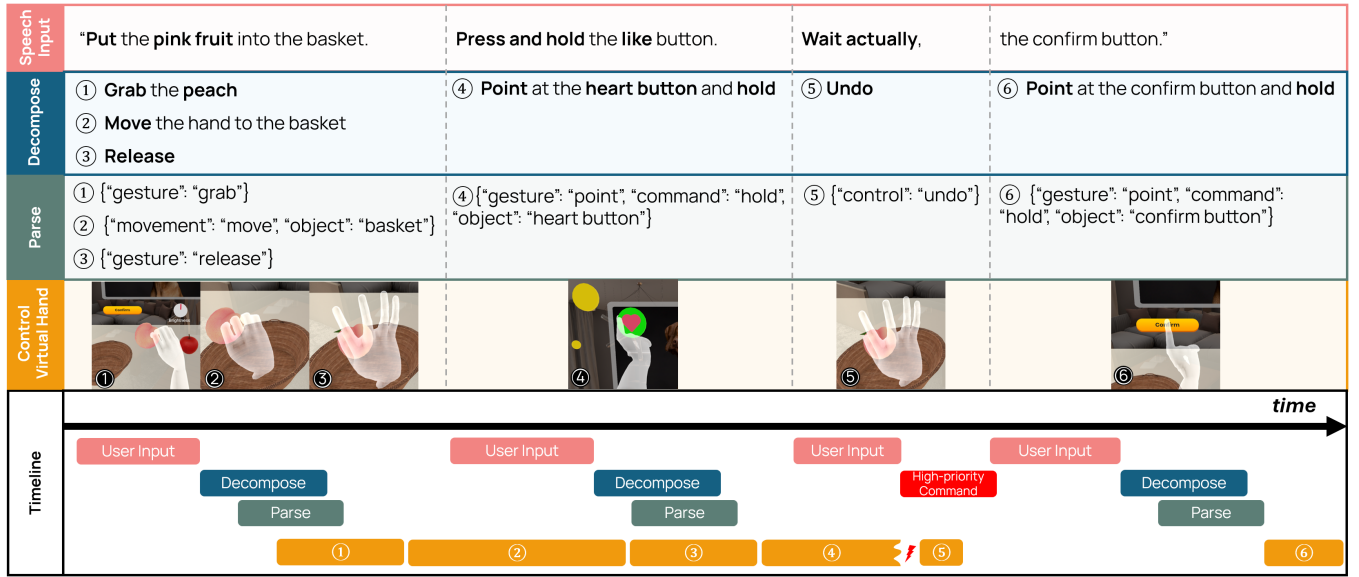


Figure 2: Example of how HandProxy would process the user’s speech commands. The user continuously talks to the system, where the system concurrently recognizes the commands, decomposes them into steps, parses them into executable hand control instructions, and calculates the sequence of hand movements to control the virtual hand performing the desired interaction. The processed instructions are added to a queue for continuous hand control. HandProxy utilizes both rule-based methods to detect high-priority commands (e.g., stop, undo) and large language model to interpret natural language input. Whenever a high-priority command is detected (e.g., (5) in the plot), HandProxy will terminate the current execution and perform the high-priority commands.

corner of a window to resize UI windows on Meta Quest, or grab and move a virtual object to place it in an desired position in mixed reality on Apple Vision Pro. Games such as *Waltz of the Wizard*¹ rely on various hand interactions, including grabbing and moving game objects, punching skulls, knocking doors, shaking hands, and tilting cups to pour water. These examples highlight the expressiveness and complexity of hand interactions beyond traditional input modalities, and demonstrate diverse interaction possibilities that could only be achieved with hands in the virtual environments.

However, users may not always be able to perform the expected hand interactions, making it challenging or even impossible to effectively interact with the virtual environments. This could be caused by situational impairments [84], e.g., when users’ hands are occupied with other physical tasks, they may not be able to manipulate virtual objects and interfaces. Environment constraints could also cause challenges, e.g., users may not be able to move freely in a confined physical space [46], and virtual objects could be out of physical reach in an enlarged virtual space [82]. Additionally, predefined hand interactions may be inaccessible to users with upper body limitations [54, 88]. These challenges indicate the need for alternative ways of interacting with virtual environments, specifically those that can reproduce or achieve similar interaction experiences and the expressiveness as the original hand interactions.

Among the various alternatives and enhancements to hand input, speech has emerged as an ideal modality due to its intuitiveness and high degree of freedom. Prior research has explored its uses

in object manipulation [79], navigation [38], and environment creation [9]. Additionally, speech has been adopted as an assistive input modality in many mainstream XR devices. For example, Apple Vision Pro provides voice control [7] as an alternative to hand input, which can initiate simple interactions, including tap, swipe, and drag and drop on 2D interfaces. Users can also issue high-level commands, such as “turn up the volume”, instead of doing it step-by-step. However, these speech interfaces typically support a limited set of hand interactions and require users to use a rigid, predefined command format to trigger actions. While this approach works for basic system-level interactions, it falls short in supporting the complex and diverse interactions a hand can do in virtual environments. This limitation is particularly relevant to interactions triggered by hand gestures or movements, such as 3D object manipulation (e.g., pinch, punch, squeeze) and physics-based interactions (e.g., slice a fruit in half with a cutting gesture). Thus, we seek to address the question: “How can we expand the affordances of speech interfaces to support expressive and diverse interactions, especially those that can be achieved via hand interactions in virtual environments?”

Instead of merely designing more speech commands, this work proposes an alternative approach: **enabling users to use natural language to control a virtual hand, where it can then simulate corresponding movements and perform necessary interactions on the user’s behalf.** This approach is motivated by the observation that many interactions result from a sequence of hand movements, where each step contributes specific meaning to the overall interaction. For example, when pulling a lever, visual effects are triggered as the hand grabs the handle, the lever moves

¹<https://www.aladin.io/>

as the hand moves, and the effect finally takes place when the hand releases it. Furthermore, many interactions are not explicitly defined but emerge as a result of other factors, such as collision and physics. For example, an application may not have a specific “clean the table” interaction defined, but the user can achieve this by sweeping the virtual hand across the table, pushing objects away through simulated collision. Therefore, it is impossible to simply create predefined speech commands for all possible interactions, as interactions themselves could be loosely defined, i.e., they are either context-dependent or dynamically generated. These considerations motivate our design choice of preserving the virtual hand within the environment. This allows users to control and reproduce actions as if real hand input is used, enabling them to perform interactions that mirror the capabilities of a physical hand.

To achieve this, we introduce HandProxy, a system that enables users to control a virtual hand through continuous, natural speech input. HandProxy is inspired by the concept of interaction proxy [92], where a virtual hand is used as the proxy layer between the speech interface and the immersive environment, as shown in Figure 1. We explored existing hand interactions in prior work, commercial devices, and VR applications [9, 35, 53], and synthesized a list of hand control primitives that could be used for decomposing and reproducing common hand interactions. Within the comprehensive space of possible hand interactions, this work starts by focusing on the fundamental, while critical, use case — one-handed interactions. Specifically, we investigate how one-handed manipulative interactions can be initiated through the speech interface and categorize them into four key primitives: gesture, target, spatial, and temporal control. These primitives are used as the foundation to reproduce various interaction, including both detailed controls (e.g., do a pinch gesture, grab the apple) or high-level interactions (e.g., maximize the volume). As shown in Figure 2, the system captures users’ natural speech, parses it into a list of executable commands with a Large Language Model (LLM), calculates the desired hand skeleton data, and renders it in the target system and application. HandProxy is optimized for real-time interaction, and we demonstrated that it can be used in a variety of interaction scenarios.

To understand how well HandProxy could perform hand manipulations from diverse natural language commands, and to gain insights from users’ experiences of controlling a virtual hand through speech, we conducted a user study with 20 participants. Our study shows that HandProxy enabled users to complete a variety of hand interactions in the virtual environment, including different types of interactions such as mid-air gestures (e.g., swipe left), direct object manipulation (e.g., twist the knob), high-level interaction tasks (e.g., increase the volume), and with varying levels of complexity (e.g., one-step to multi-step interactions). Participants reached a 100% task completion rate, and took an average of 1.09 attempts for their speech commands to be correctly executed by the system (with a command execution accuracy of 91.8%). HandProxy was able to handle diverse variations of participants’ commands for the same tasks, such as using descriptive commands (e.g., “touch your index finger and thumb” to pinch, “grab the red fruit” to grab an apple), varying levels of details (e.g., to increase the volume, either grab the slider and move up, or directly say “maximize the volume”), or sentence structures. Furthermore, participants reported the system to be intuitive, effective, and require minimal learning to use,

and pointed out possible improvements including more detailed feedback for enhanced disambiguation and discoverability, greater responsiveness, and supporting additional hand controls.

The specific contributions of our work therefore include:

- (1) A set of primitives to categorize common hand interactions in the virtual environment, allowing hand interactions to be decomposed and reproduced.
- (2) A real-time system, HandProxy, that enables users to issue natural speech commands to control a virtual hand to simulate and perform hand-based interactions in the virtual environment.
- (3) An investigation into the effectiveness and user experience of using speech to control hand movement for various interaction tasks in the virtual environment.

2 Related Work

Our work is based on the literature of hand gesture input in XR, alternative input modalities for hand interactions, speech interfaces, and interaction proxies.

2.1 Enhancements and Alternatives for Hand Interactions

Hand interactions have been widely used as an input method for interacting with virtual environments as it is intuitive, expressive, and preferred by users [16, 75]. These advantages have made them a core interaction technique across various devices and application domains, including object manipulation [34, 62, 65, 66], virtual collaboration [70], creativity support [14, 44, 80], gaming [1], and object retrieval [60].

Despite these advantages, hand interactions also come with certain limitations. Situational impairments [84] such as occupied hands in AR, physical disabilities [21, 86, 88] such as upper body limitations, fatigue caused by long-term use of mid-air gestures [37, 43], or environment constraints such as the confined physical space [46] or out-of-reach virtual objects [82] could limit users’ abilities to perform hand input. Additionally, users may prefer different gesture [61, 63, 85], making the standardized gesture input less desirable in certain situations.

To address these challenges, researchers have explored various enhancements and alternative interaction techniques. Some approaches expand hand input capabilities, such as extending the virtual hand for reaching distant objects [64, 82], using multiple virtual hand copies for easier object selection [68], and exploring customized gesture sets for users with motor impairments [73]. Others focus on alternative input modalities, including sensor-based modalities [87, 88], speech-based commands [51], and multi-modal interaction techniques, such as gaze + voice [31], hand + gaze [78], and hand + speech [83]. However, they are either designed to address a limited scope of specific scenarios (e.g., basic object manipulation, locomotion) or may require complex input setups, which are less practical to be deployed on existing devices for a broad range of interaction scenarios.

Building on insights from prior research, HandProxy uses speech as an alternative modality. This choice is motivated by the fact that speech is widely accessible across various devices and is already integrated as a built-in control mechanism in many commercial

systems [7, 8]. We explore how speech can be used to initiate an expressive and diverse range of interactions that usually require hand input. Rather than simply substituting individual gestures for speech commands, our work expands the capabilities of the existing speech interface through a proxy hand, enabling users to perform hand-based tasks through flexible and intuitive spoken input.

2.2 Speech Interfaces in Immersive Environments

Speech has been widely adopted as a control mechanism across various scenarios. For example, users can issue supported voice commands to interact with mobile devices [12, 77] or desktop user interfaces [13, 41]. In robotics, speech is used to control robot actions [40, 72]. More recently, the integration of speech interfaces with large language models (LLMs) has expanded these capabilities [11, 24, 28, 59, 67, 76, 94], enabling functions such as motion planning, task decomposition, and flexible natural language input.

In immersive environments, speech interfaces have been shown to be particularly intuitive, expressive, and natural due to their high degree of freedom [15, 35, 51]. As a result, it has emerged as a promising input modality for interactions in virtual environments. Prior work has explored speech-based interactions for various tasks, including virtual objects manipulation [15, 18, 53, 69], locomotion [38], and scene creation [9, 91], using verbal or non-verbal commands such as breathing or sound actions [2, 71, 95]. Commercial XR devices have also integrated speech as a built-in assistive input modality. For example, Apple Vision Pro [7], Meta Quest [5], and Microsoft HoloLens [8] allow users to issue system commands (e.g., volume adjustment, power control) or perform simple hand gestures (e.g., tap, swipe, drag-drop) using speech. While these approaches demonstrate the versatility of speech interfaces, they require users to follow rigid command structures, and the interactions supported are limited to basic hand gestures and system functions.

Recently, integrating large language models (LLMs) into speech interfaces has opened up diverse interaction possibilities, allowing users to directly create, manipulate, query, and engage with their environments through flexible speech input [9, 25, 79]. Building upon these prior works, we specifically explore ways to expand the affordance of speech interfaces through a generalizable approach, so that it can enable more interaction possibilities, while requiring minimal direct modification of individual applications to support it. To achieve this, we introduce a virtual proxy hand in the pipeline. By describing the movement of a proxy virtual hand in flexible speech input, we seek to enable expressive interactions in virtual environments, especially those that hands or only hands can do. Additionally, to design an effective speech-based system, we built upon prior work on speech interfaces [35, 56, 59], user expectations for AI-driven agents in VR [9], and contextual considerations in speech systems [90]. These insights inspire the design choices of HandProxy to incorporate embodied knowledge, contextual understanding, conversational memory, interaction state control, and common knowledge integration to enhance the system's ability to interpret and execute user commands effectively. Combining these design choices, our system can effectively support users to describe their interaction intentions, and the proxy hand can then perform the necessary interactions on the user's behalf.

2.3 Interaction Proxies

Interaction proxies are the extra layer inserted between the original and the manifest interface to add or modify interactions without changing the app's source code [92]. Their ability to introduce new functionality with minimal modifications makes them particularly valuable for tasks such as input remapping [48, 50, 92, 93] and UI automation [47]. This concept has also been applied to immersive environments. For example, researchers have created tangible proxies that map physical input with digital interactions [17, 23, 27, 36, 42]. Other work explores remapping of complex 3D input motions in VR to a more accessible ranges of motion or simpler input devices [81]. Some work specifically explored the remapping of hand interactions through the proxied interface in virtual environments, such as remapping VR hand interactions to mobile devices [46] or finger movement [74] to facilitate interactions in constrained spaces. Additionally, McGlashan et al. [51] introduced proxy agents as an interaction metaphor in VR, allowing users to issue commands to virtual agents that execute tasks on their behalf.

Inspired by these ideas, HandProxy explores how expressive hand interactions can be effectively translated into speech commands. Rather than relying on a rigid set of predefined voice commands for each interaction, HandProxy allows users to control a virtual hand as a proxy, enabling it to perform actions just as a real hand would. This approach ensures that users can issue commands in a natural and intuitive way, while the virtual hand continues to interact with applications as expected. Also, it reduces the need for system-specific modifications to support the mapping and functionality of the speech interface, making this approach compatible and generalizable across different applications that accept hand input.

3 Primitives of Hand Interaction

To effectively perform interactions using the virtual proxy hand, it is essential to define a control framework that can reproduce a wide range of hand movements. Anatomically, the human hand has 27 degrees of freedom (DoF), covering finger extension, flexion, abduction, adduction, wrist rotation and translation [10]. Given this complexity, directly replicating all possible hand movements would be impractical for virtual interactions, as each of the 27 DoF needs to be mapped to an input for control. Therefore, in this section, we introduce a simplified control framework designed to balance usability and expressiveness, and ensure that commonly used hand interactions in virtual environments can be effectively reproduced while maintaining intuitive control.

Our framework is inspired by the concept of hierarchical gestures [26, 49], where complex hand interactions can be formed by strategically combining multiple primitive gestures. However, in this work, we take a slightly different approach — instead of building up interactions from simpler gestures, we explore how commonly used hand interactions can be decomposed into a small set of shared fundamental control primitives. The goal of this decomposition is to create a compact yet expressive control framework that simplifies the reconstruction of hand interactions while still accommodating a wide range of interaction possibilities. Additionally, this approach could align more naturally with speech input, which is inherently structured as a combination of smaller linguistic units (i.e., words).

For example, given the command “pinch up”, it can naturally be decomposed into a gesture of pinch and a movement of up, matching the primitive hand controls. With these characteristics, the framework simplifies the integration of hand control with speech input while also making interactions intuitive for users.

To develop this framework, we reference prior work on hand gesture definitions [45], mid-air gestures [39, 61, 63], gesture design considerations [85], as well as gesture vocabularies in XR device control [3, 4, 6] and XR applications [1]. In this work, we focus on the fundamental and critical use case — one-handed manipulative interaction, and identified a set of fundamental hand control “building blocks” that served as the core components for reproducing diverse interactions, including *gesture control*, *target control*, *spatial control*, and *temporal control*, which are discussed below.

Gesture Control: Gesture control defines the fundamental static or dynamic gesture required to initiate a specific interaction, independent of spatial or temporal constraints. For example, to twist an object clockwise, the hand must first assume a grab gesture, then perform a rotation to complete the action. These gesture units build the foundation of a hand interaction. In this work, we demonstrate a core set of base gestures commonly used in virtual environment interactions, such as grab, pinch, and cut. For further technical details on extending the gesture set, please refer to [section 4](#).

Target Control: A gesture can be either object-independent or object-dependent. For example, a user may perform a mid-air pinch gesture without interacting with any object (object-independent), or they may execute the same gesture on the corner of a virtual interface to manipulate it (object-dependent). The goal of target control is to determine whether a gesture depends on an object, identify the specific target the user is referring to, and adjust the hand movement accordingly to ensure proper interaction. For example, given the command “grab the rightmost watermelon”, the target control should recognize this as an object-dependent action, resolve “the rightmost watermelon” as a specific object, and adjust the virtual hand’s movement path to ensure it reaches and successfully grabs the correct target.

Spatial Control: Spatial control determines the movement and rotation of the hand in addition to the previous controls. This includes the movement of the hand in the 3D virtual environment and the rotation of the hand, including pan left/right, roll left/right, and tilt forward/backward. Beyond basic gesture and target control, spatial control modifies the movement path and behavior of the hand to enable more complex interactions. This allows users to perform tasks such as placing an object at a specific location, grabbing and moving a virtual slider, twisting a knob, or rotating the hand to inspect an item from different angles.

Temporal Control: Temporal control acts as a playback control, regulating the timing and execution of hand interactions. This includes the ability to pause, resume, accelerate, decelerate an ongoing interaction, as well as undo, redo, or repeat a previous interaction. These controls enable interactions that depend on timing and states. For example, a user may pull the lever and stop pulling when they heard a audio cue to stop, or resize the window by pinching and moving to the right until satisfied. Users can also correct an undesired interaction through undo, or simplify repetitive interactions through repeating a specific hand state (e.g., do it again / 10 times).

Given these control primitives, a hand interaction can be decomposed into one or more combinations of these foundational elements. As shown in [Figure 3](#), “pull up” can be broken down into a grab gesture (Gesture Control) and an upward movement (Spatial Control). High-level commands that require multiple steps can be decomposed into multiple combinations of these primitives. For instance, “put the apple into the basket” involves three steps – grab the apple, move to the basket, and release it. Through this structured decomposition, the system can effectively interpret and reproduce a wide array of hand interactions, accommodating both simple and more complex interactions. This framework guides our system design as detailed in [section 4](#), ensuring that our approach remains flexible and intuitive to various interaction needs.

4 HandProxy: A Speech System for Virtual Environment Interaction Using a Proxy Hand

HandProxy is a system that converts a user’s live speech commands into dynamic virtual hand movements, allowing the proxy hand to interact with the virtual environment on the user’s behalf. Overall, the HandProxy system has the following key capabilities:

- (1) **Flexible natural speech commands:** HandProxy allows users to describe interactions in their own words without rigid formats. It interprets input, decomposes complex commands into executable steps, and prompts for clarification if needed. Users can issue commands at different levels of detail—from precise hand movements (e.g., “pinch” or “grab the box”) to high-level goals (e.g., “increase the brightness” when a brightness knob is present), which brings additional flexibility for user input.
- (2) **Context-aware command processing:** By leveraging environment metadata and real-time object positions, HandProxy interprets contextual references, allowing users to specify targets based on relative positions (e.g., “grab the watermelon in the middle”) or object attributes (e.g., “pick up the pink fruit”). It also maintains a history of interactions, enabling users to reference past actions (e.g., “do it again”) or use undo/redo functions to restore a hand state, to support intuitive and efficient interaction.
- (3) **Real-time streaming input and execution:** HandProxy takes speech input continuously, allowing users to speak naturally while the system interprets and executes commands in parallel for a seamless interaction experience. High priority commands (e.g., “stop”, “undo”) are processed instantly via rule-based methods, while LLMs handle complex instructions. System components run in parallel to minimize delays.
- (4) **Feedback for disambiguation and system transparency:** To enhance clarity and transparency, HandProxy provides visual feedback overlays showing recognized commands, expected hand movements, and prompts for retry when needed. When multiple objects match the description in the command, it shows disambiguation labels to clarify user intent.

Below, we begin with an overview of the system design elements, followed by technical details of each components, and conclude with a summary of the design iterations conducted during development to offer rationale for our design choices.

Primitives	Description	Example decompositions						
		Pinch	Pull up	Punch the bag twice	Twist the knob to the right	Put the apple into the basket (multi-steps, each step per column)		
Gesture control	Defines the base gesture	Pinch	Grab	Punch	Grab	Grab	-	Release
Target control	Controls object dependency	-	-	Bag	Knob	Apple	-	-
Spatial control	Modifies hand movement and rotation	-	Movement, up	-	Rotation, pan right	-	Movement, to the basket	-
Temporal control	Controls hand movement playback	-	Hold	Repeat, 2 times	Hold	Hold		-

Figure 3: Examples of how hand interactions can be decomposed into one or multiple combinations of hand control primitives.

4.1 Design Elements

As illustrated in Figure 4, HandProxy consists of three major components: speech understanding, hand control, and visualization & feedback. The speech understanding module continuously listens to and processes user input in a streaming fashion in real-time using Google Speech-to-text API [33]. It segments long speech inputs into individual commands based on punctuation, and adds them to the command queue. Using GPT-4o [58], the system then interprets each command and decomposes it into a sequence of instructions in the json format. Once instructions are available, the hand control module uses the hand pose sequences that was either retrieved from FPHAB [30] dataset or recorded using LeapMotion and modifies the skeleton data to accommodate movement, rotation, and hand-object interaction. The playback of the hand data is controlled by the hand state manager to support interaction timing control, and is streamed to the target virtual environment for virtual hand control, along with additional visualizations and feedback to be overlaid on top of the user's view. The system components run in parallel to increase the efficiency and avoid blocking the hand movement and impacting user experience.

The HandProxy system is implemented in Python, with its output streamed to a Unity application via the TCP connection. The Unity application renders a sample 3D virtual environment that contains interactable objects and a virtual hand, simulating an immersive application that supports hand gesture input. It overlays visual feedback specific to HandProxy on top of the immersive environment, as shown in Figure 5. For demonstration and prototyping purposes, the Unity app runs on a laptop, but can be deployed to other supported platforms (e.g., Android on Meta Quest). Demonstrations of the interaction experience on desktop and XR headset setups are provided in the supplementary video.

4.2 Speech Understanding

The speech understanding module recognizes and interprets the user's command. It determines whether the speech command is an interaction command or a random speech, checks if it is ambiguous or interpretable, and decomposes it into structured, system-interpretable instructions for the hand control module to execute.

For speech recognition, HandProxy uses the Google Speech-to-Text API, which is configured to return intermediate results for real-time streaming recognition. The speech input is segmented into

individual sentences, determined by terminal punctuation on the final results returned from the API. Each sentence is then treated as a command and is added to the command queue for interpretation.

For command interpretation, HandProxy uses a multi-level approach for command processing. Similar to the reactive and deliberative layers in robotics [57], the system uses keyword matching and directly responds to high-priority commands, with LLM-based processing for more complex inputs. Specifically, as soon as intermediate results are received from the speech recognition service, HandProxy checks predefined keywords and synonyms related to temporal controls as discussed in section 3. These commands — such as “stop,” “hold,” or “undo” — are time-sensitive and need to be executed with minimal delay. If a matching is detected, the system triggers the corresponding temporal control immediately. Otherwise, the system retrieves the earliest available command from the command queue and processes it using GPT-4o. The model is guided by a structured system prompt and is provided with a list of supported control categories from section 3, which is listed below:

```
Your objective is to break down a user command into actionable
control components that the system can execute. Once you receive
the command, follow these steps meticulously:
#### **Step 1: Command Fixing**
1. Input command is converted from speech, so minor errors may
occur. If you notice any, please correct them and replace the
original command with the fixed version.
#### **Step 2: Components Matching**
1. Carefully read through the entire command.
2. Imagine you are a person with only your right hand available
(currently empty), standing in a wide space, surrounded by the
objects and environment described in the command.
3. With point 2 in mind, align each command part (and necessary
actions for execution, e.g., grabbing an object before moving it)
with a corresponding control component listed below. Users may
say or describe the interaction in various ways, the system should
match the similar concept together. For example, "chop" and "cut"
should both be matched to "cut", etc.
4. If no components can be matched, proceed DIRECTLY to Step 4.2,
skip intermediate steps.
#### **Step 3: Ensuring Correct Order**
1. Components should follow the user's expected execution sequence.
#### **Step 4: Structuring Output**
1. Once the components are matched, fill in all fields within
each component. ENSURE ALL FIELD VALUES ARE SELECTED FROM THE
PREDEFINED OPTIONS PROVIDED BELOW.
2. Format each component as a JSON object as defined below, then
output them sequentially in an array. If no components are matched,
set the value of components to an empty array.
[...# a list of available controls and parameters for each
component are appended after the prompt]
```

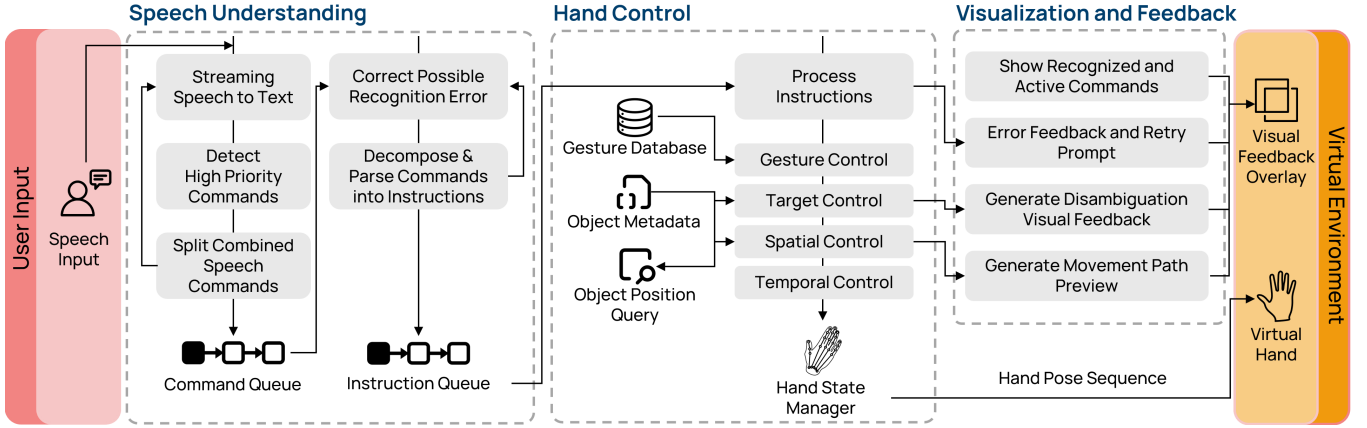


Figure 4: An overview of the HandProxy system structure. Given the user input, the speech understanding module recognizes the speech into commands, and parses the commands into system-interpretable instructions. The instructions are then sent to the hand control module, which generates the corresponding sequence of hand pose to control the movement of the virtual hand. Visualization and feedback module generates appropriate visual feedback for disambiguation and better transparency.

The LLM is used to (i) correct potential speech recognition errors (e.g., power bottom → power button), as used in other works like [89], (ii) decompose high-level commands, if any, into executable steps (e.g., put the cube into the basket → grab the cube, move to the basket, and release), and (iii) parse individual steps as a list of instructions as json. Also, since people tend to use a surprisingly great variety of words to refer to the same thing [29], the system prompt instructs the model to generalize beyond exact words provided in the supported instructions to accommodate diverse ways users may phrase their commands. For example, commands with similar intent are mapped to the same control action (e.g., “pinch” = “tap index and thumb,” “pink circular fruit” = “peach”). If a given input is not relevant — meaning it does not appear to be an interaction command — the model is instructed to ignore it and return an empty list instead of attempting to interpret unrelated speech. Additionally, user commands are saved within a command list, allowing the system to refer back to previous inputs when necessary (e.g., resolving references to earlier commands).

This approach ensures that the LLM interprets user commands and maps them to the appropriate hand interactions, including:

- **Gesture Control:** [“pinch”, “point”, “push”, “grab”, “swipe”, “punch”, “squeeze”, “cut”, “thumb_up”, “thumb_down”, “open_hand” (i.e., release)]
- **Target Control:** (i) A list of interactable object names in the virtual environment, their current positions, and a list of descriptive tags for each object, (ii) supported relative constraints to identify objects: [“below”, “above”, “to the left of”, “to the right of”, “in front of”, “behind”, “closest”, “farthest”, “first”, “last”, “on the left”, “in the middle”, “on the right”]
- **Spatial Control:** (i) translational control: [“up”, “down”, “left”, “right”, “forward”, “backward”], (ii) positional control: [“on top of”, “under”, “in front of”, “behind”, “to the left of”, “to the right of”], (iii) rotational control: [“pan left”, “pan right”, “roll left”, “roll right”, “tilt up”, “tilt down”]

- **Temporal Control:** [“stop”, “continue”, “faster”, “slower”, “undo_step”, “redo_step”, “hold”]

At the end, the speech understanding module outputs a json object following this format. For example, given the command “pinch the cube,” the system generates the following instruction:

```
{["component_type": "gesture", "value": {"gesture_type": "pinch",
"object": "cube", "is_ambiguous": False}]}
```

Here, the system identifies a gesture control action, which specifies a pinch gesture on the cube object. If only one cube exists in the environment, the `is_ambiguous` variable is set to `False`. However, if multiple cubes are detected, it is set to `True`, triggering a disambiguation mechanism and corresponding visual feedback in other modules to prompt the user for clarification.

For more complex commands, the LLM references both the list of supported actions and its general knowledge to determine the best way to combine supported hand controls to finish the user’s request. For instance, given the command “peach into the basket,” the system generates multiple sequential steps:

```
{["component_type": "gesture", "value": {"gesture_type": "grab",
"object": "peach", "is_ambiguous": False}, {"component_type":
"movement", "value": {"movement_type": "translational", "ob-
ject": "basket", "is_ambiguous": False, "position": "on top of"}},
{"component_type": "gesture", "value": "release"}]}
```

By dynamically generating one or more executable steps, HandProxy lets users issue both low-level direct commands and high-level intent-based commands, which provides an adaptive and natural interaction experience.

4.3 Hand Control

The hand control module executes instructions from the json to generate a sequence of hand pose data to control the virtual hand. It maintains a hand state manager, which tracks key hand properties such as the current hand pose, wrist position, playback status (e.g., whether an action is in progress), and the sequence of upcoming hand poses. When a new instruction arrives, the hand control

module activates the relevant control sub-modules discussed in [section 3](#) to update the hand pose data. Next, we describe each control module in detail.

4.3.1 Gesture Control. The gesture control module generates a sequence of hand pose data to represent the required gesture, which are retrieved from the FPHAB [30] dataset or captured using Leap-Motion, but is extensible to gesture datasets that include the 2.5D coordinates (x_ratio , y_ratio , z_depth) of 21 hand joints. Currently supported gestures include grab, pinch, point, push, swipe, cut, punch, squeeze, thumb up/down, and open hand (release).

To enrich data with additional gesture information, each gesture is manually annotated with details about the gesture's object dependency and different gesture stages [52] that could benefit the gesture control. The full metadata format can be found in [Appendix A](#), and some key attributes include:

- (1) **Is static:** Specifies whether the gesture is static or dynamic.
- (2) **Interacting frame:** Identifies the frame at which the hand fully interacts with an object (if applicable).
- (3) **Interacting joint:** Specifies the joint primarily involved in the interaction (e.g., the index fingertip for a pointing gesture). If unspecified, the wrist is used by default. This information helps hand movement calculations during object interactions to better match the expected hand behavior.
- (4) **Segments:** Defines the start and end frames for different gesture phases, including preparation, stroke, and retraction, to support additional features such as gesture holding.

Combining all the data together, when given a gesture command, the module loads the sequence of hand pose data from the gesture database. Downstream modules can use the pose data at each frame as an 21×3 matrix, modify it, and send it to the virtual environment.

4.3.2 Target Control. The target control module handles object-dependent hand interactions by identifying and aligning the virtual hand with the intended target. It first resolves the target object using positional constraints, such as “below,” “above,” “to the left of,” “to the right of,” “in front of,” “behind,” “closest,” “farthest,” “first,” “last,” “on the left,” “in the middle,” and “on the right.” These constraints are resolved by retrieving the current positions of all virtual objects through a query to Unity application and calculating their relative spatial relationships.

Once the target object has been determined, the module calculates a path to move the hand from its current position to the target object. It first determines the total distance to move, calculates the movement step size by dividing the total distance by the number of frames for this gesture, and cumulatively add step sizes to original hand pose data for all the following gesture frames. This calculation also accounts for the interacting joint (as discussed in Gesture Control) to ensure proper alignment of the hand with the object. For example, if the command is to point at a button, the system uses the interacting joint defined in the gesture metadata to set the index fingertip to align with the center of the button when calculating the movement path. By dynamically modifying the original hand pose sequence, the target control module ensures that gestures interact with objects in the correct fashion.

4.3.3 Spatial Control. The spatial control module manages the movement and rotation of the virtual hand. It supports translational

movements in six directions: “up,” “down,” “left,” “right,” “forward,” and “backward.” Additionally, it enables rotational adjustments, including “pan left,” “pan right,” “roll left,” “roll right,” “tilt up,” and “tilt down.” Specifically, translational movements are applied by adding a speed vector to the hand pose data, while rotational adjustments are calculated by rotating the joint coordinates along the estimated hand axis using the vector of middle metacarpophalangeal joint (MCP) to index MCP, middle MCP to wrist, and the vector perpendicular to these two vectors.

Similar to target control, movement commands can also reference target objects along with relative position constraints such as “on top of,” “under,” “in front of,” “behind,” “to the left of,” and “to the right of.” In such cases, the system computes the movement path similarly to object interactions but applies an offset to reach the specified position relative to the target object.

4.3.4 Temporal Control. Temporal control specifies the timing and state changes of hand interaction, including “stop,” “continue,” “faster,” “slower,” “undo_step,” “redo_step,” and “hold.” Hand pose data from earlier stages is sent to the hand state manager, which keeps a playback speed (frames per second, fps) and playback state (stop, play) that can be adjusted based on temporal control commands. For the hold command, specifically for holding a specific gesture (e.g., as in grab and hold the cube), the temporal control adds a key frame at the interacting frame (as defined in gesture control) and pauses the playback at that frame to hold the gesture.

Additionally, the hand state manager maintains a history of hand poses after each command. This enables “undo” and “redo” functionality, allowing users to revert to previous hand states or redo an action if needed. If an error occurs, the system can quickly restore the last hand state, providing a flexible interaction experience.

At the end, the hand control module outputs the hand pose for the current frame as a list of 63 numbers ($21 \text{ joints} \times (x, y, z)$), and sends it to the Unity app to control the virtual hand.

4.4 Visualization and Feedback

To enhance system transparency and reduce ambiguity, HandProxy uses various visual feedback mechanisms that are overlaid on the user's viewport. In the prototype Unity app, visualization is implemented as a canvas overlaid on the main camera view. Each visualization is triggered when it receives the command from the HandProxy. [Figure 5](#) shows examples of each feedback type. To ensure users understand how their commands are processed, the system displays both the recognized (full input) and active (which part of the input is being executed) commands. If a command is irrelevant (e.g., not a valid hand interaction) or cannot be interpreted, the system highlights an error message in the recognized input and prompts the user to rephrase the command. For cases where the user refers to an ambiguous object (i.e., multiple similar objects exist in the environment), the system provides disambiguation hints by overlaying numbered labels on each possible target, allowing users to specify the object by its assigned number. To further increase transparency in hand movement execution, the system visualizes the movement path. As shown in [Figure 5d](#), a sequence of arrows appears along the trajectory, indicating the direction in which the hand is moving. If the command involves multiple steps, the full movement sequence is previewed to give users an overview of how

the hand will execute the task. These visualizations provide users with greater visibility of the system to bridge the gulf of evaluation, improving both command accuracy and interaction efficiency while reducing errors and misinterpretations.

4.5 System Latency

The system latency is composed of two main components: speech recognition and processing latency, and command processing latency. Speech recognition and processing latency measures the time from just before an audio chunk is sent via an API request, to when the recognized text is split into commands. Based on 100 requests, the average latency for this process was 0.18 seconds, with a standard deviation of 0.07s. The command processing time is the time it takes for the system to interpret the command and generate a sequence of corresponding hand pose keypoints. We measured such latency for both commands that matched high-priority keywords, and those processed by LLMs. Through a test of 100 commands that matched the high priority keywords, all commands were executed within 0.001 seconds. Through a test of 100 requests that included both simple (e.g., requires one step), complex (e.g., needs to be decomposed into multiple steps), and irrelevant commands (e.g., not a valid interaction instruction), the average response time was 1.39 seconds, with a standard deviation of 0.81s.

4.6 Adapting to New Gestures and Environments

HandProxy can be extended with additional gestures. It accepts hand pose data in the format of a sequence of 21 keypoint coordinates (x, y, z) and a gesture metadata in the format shown in [Appendix A](#). The widely used 21-joint format enables data from various sources to be easily integrated, such as hand pose datasets [30], recorded data (e.g., from LeapMotion or MediaPipe), and hand pose generation models [22], and HandProxy automatically normalizes the data. Existing built-in controls, such as target, spatial, and temporal controls and other gestures, are still compatible with the newly added gestures.

To adapt to new environments, HandProxy requires the name and optionally descriptive tags of interactable objects, similar to the accessibility metadata for UI elements on mobile [32] or web interfaces (e.g., DOM tree [55]). The GPT prompt is dynamically updated with the object information to process new commands. Note that due to limited direct access of the hand input on XR systems, in our current implementation, the virtual hand is added as an object within the application. However, we envision that with system-level support, HandProxy can directly control the virtual hand input of the operating system and retrieve interactable object information to achieve a more unified, cross-application control.

4.7 Design Iterations

The HandProxy system was iteratively designed and developed through pilot studies with 13 participants recruited from the student group at the authors' institution. Participants were asked to use HandProxy to perform object selection, manipulation, and transformation tasks in a test virtual environment, and is then interviewed for their experiences and potential improvement suggestions. Here, we summarize key aspects from these iterations, with the goal

of providing additional rationale behind our design choices and insights into user preferences.

4.7.1 Enhancing the Speech Interface. The speech interface is a core component of HandProxy. In our initial design, we followed a traditional turn-based interaction approach, similar to off-the-shelf speech interfaces [7], where users issue a command, wait for execution, and then issue the next command. However, our pilot studies revealed that this approach was inefficient for hand control. Many participants preferred to issue multiple commands continuously, describing an entire sequence of actions in one go. The forced pauses between commands disrupted the interaction flow, making it less natural. Additionally, it was difficult for users to issue commands to override current executions, such as stop or undo, as users had to wait for the previous command to finish before issuing a correction. As a result, we redesigned the speech interface to support continuous input, allowing simultaneous speech recognition and action execution. This change also motivated the parallelized system architecture, where each module runs in a separate thread, ensuring that speech processing, command interpretation, and execution run without blocking other processes.

Additionally, we observed variations in user preferences for the level of detail in commands. Some preferred step-by-step instructions, such as “move left,” “grab,” while others issued higher-level commands, such as “put the cube into the basket,” expecting the system to infer intermediate steps. This motivates the choice of integrating LLMs (GPT-4o) than pre-defined commands to support flexible and diverse speech input and leverage their reasoning capabilities to interpret user intent dynamically.

4.7.2 Integrating Contextual Information. Throughout our iterations, we observed that participants naturally used descriptive references to identify target objects, including shape, appearance, and relative location. Examples include “green ball” or “the object next to the button.” This demonstrated the need for contextual understanding, which motivated our decision to incorporate object metadata into the system.

While advanced vision models could enhance contextual understanding, we drew inspiration from accessibility metadata in 2D user interfaces and implemented a metadata structure for virtual objects. Each object is assigned a set of descriptive tags, allowing the system to recognize and differentiate objects based on user descriptions. Additionally, many object references rely on common knowledge (e.g., “a green fruit with stripes” is likely a watermelon). To enhance generalizability, we integrated LLMs to allow HandProxy to reason beyond predefined metadata and infer likely object references based on common knowledge and context.

4.7.3 Improving System Transparency. Speech recognition errors are a common challenge, and users wanted to know how their commands were interpreted. This feedback motivated several transparency focused design choices, including showing the recognized results and visualizing the path of the expected hand movement. These features ensure that users can intervene in real-time when error happens using commands like stop or redo, which ultimately improves interaction reliability and usability.

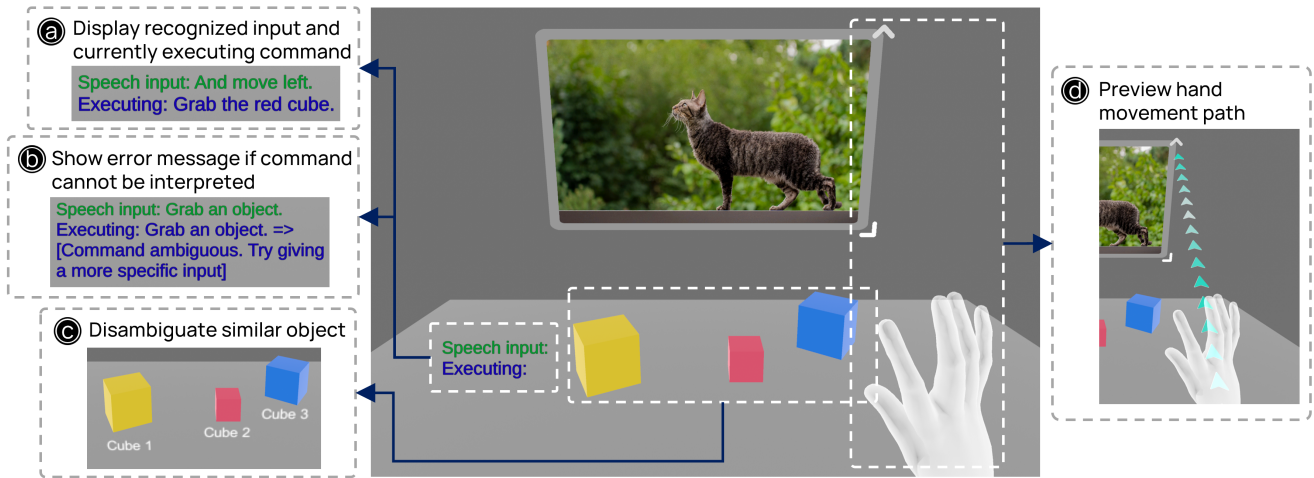


Figure 5: Examples of system feedback: (a) an overlay streaming speech recognition results and currently executing command, (b) an error message indicating the command is ambiguous, prompting the user to try again, (c) a disambiguation overlay that labels ambiguous objects, where the user can clarify the desired one by saying its number, and (d) a preview of the hand movement path, shown as a list of arrows pointing towards the destination.

5 User Study

We conducted a user study with 20 participants to evaluate the HandProxy system and gather insights on usage patterns and feedback. Specifically, our research questions are as follows:

- RQ1.** To what extent can participants complete a wide variety of tasks using HandProxy?
- RQ2.** How effectively can the system interpret the diverse ways in which users issue commands?
- RQ3.** What strategies, preferences, and methods do participants use for specifying intent?
- RQ4.** How do participants experience and perceive HandProxy?
- RQ5.** What other expectations do participants have for the system?

5.1 Participants

We recruited 20 participants (9 female, 11 male, age 18-31) from the student population at our institution. Based on a self-reported survey, 3 participants had no prior experience in immersive environments (e.g., VR/AR), 13 were beginners, and 4 were intermediate users. For experience with hand-gesture-controlled interfaces (e.g., XR headset like Meta Quest or hand tracking cameras like Kinect), 5 participants reported no experience, 12 were beginners, and 3 were intermediate users. Regarding speech interfaces, 1 participant had no experience, 5 were beginners, 12 were intermediate users, and 2 were experts. Examples of speech systems included virtual assistants like Siri or Alexa, ChatGPT voice mode, and speech input on TV remotes. The study was approved by the IRB, and participants received \$30 gift cards as compensation.

5.2 Apparatus and Procedure

The study sessions were held in a small meeting room. Participants sat in front of a TV connected to a workstation laptop with i9-12900H CPU, which runs the 3D virtual environment in Unity. This setup is to provide an equivalent experience for all participants. As shown in Figure 6, the virtual environment in Unity is designed to have different 3D objects and UI widgets, supporting a variety of

interaction tasks. The interactions are triggered only by collision or gesture detection of the virtual hand. The Unity app does not provide API to trigger interaction. Each study session took 2 hours.

Overview (10 minutes): participants were broadly introduced to the project and the system setup, and were asked to fill out the pre-survey on their prior experiences.

Study tasks (60 minutes): for each task, participants first watched a video clip demonstrating a virtual hand performing the intended hand interaction. They were then instructed to replicate the interaction using speech through HandProxy. To minimize bias toward specific commands, no guidance was provided beyond the video demonstration, except when participants required clarification or assistance. For mid-air gesture tasks, participants were asked to complete the task directly. For other tasks, interactions were conducted with three different objects. During the initial *practice session*, participants used the first object to explore the system's capabilities and limitations by experimenting with different commands. In the subsequent *test session*, they performed the same interaction on two additional objects, aiming to complete the task as accurately as possible. The tasks below are selected based on common types of interaction tasks in the virtual environment [53], including selection, manipulation, and transformation.

- (1) *Warm-up with mid-air gestures:* participants were asked to use speech to reproduce common mid-air gestures on commercial XR devices [3, 6] as a warm up, which included pinch, swipe left, double pinch, and thumb up. They served to familiarize participants with the system and the basic gestures, and to prepare and on-board them for subsequent tasks.
- (2) *Hand interactions:* participants were asked to perform a wide range of interaction tasks for virtual environment [53], including object selection, manipulation, transformation. The tasks included object dependent gestures (grab {apple, peach, blue cube}, press {confirm, minimize, power} button, pinch {blue cube, volume slider, resize button}, push {confirm, like,

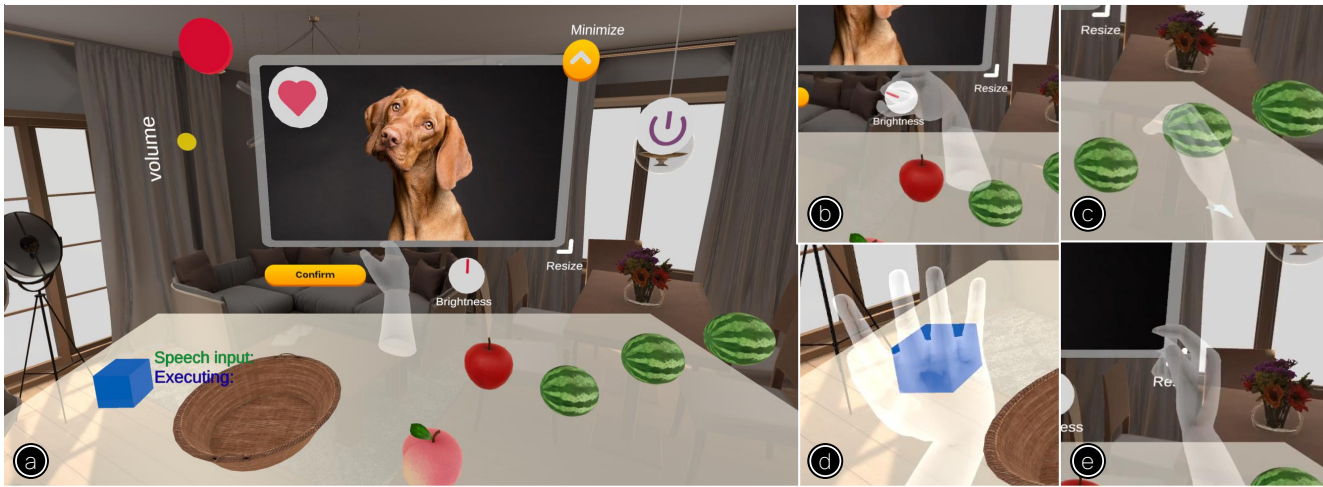


Figure 6: Virtual environment for the user study (a), with different interactable 3D objects and UI widgets. Participants were asked to perform various hand interactions by commanding the virtual hand in the environment. Task examples include but not limited to twist knob (b), grab fruit (c), push cube (d), and pinch to resize window (e).

power)) and disambiguation (grab the {left, middle, right} watermelon among the 3 watermelons).

- (3) *Movement and rotation control*: this included hand movement (move {left, up, forward}), object movement (put the {apple, peach, blue cube} into the basket, rotation (turn the brightness knob {clockwise, counterclockwise}). Participants were also asked to perform tasks that required timing control. This included grab the {apple, peach, cube} and move to the left according to a light signal (green = move, red = stop), and to press and hold the {power, like, confirm} button.
- (4) *Complex tasks*: for these tasks participants were only shown the image of the final outcome, without demonstrating the steps to accomplish it. It was up to the participants to decide what and how many steps to take to finish the task. These include tasks to make the window wider, put apple, peach and first watermelon into the basket, and maximize the volume.

Free exploration (10 minutes): participants had the opportunity to freely explore the environment using commands of their choice to investigate the system's capabilities and limitations. A Likert scale questionnaire on system usability was given at the end.

Semi-structured interview (40 minutes): participants were interviewed about system performance, user experience, interaction strategies, desired features, and suggested improvements.

5.3 Methodology and Procedure of Data Collection and Analysis

To support quantitative analysis, the system logged the recognized text, decomposed json output, executed hand controls, and system feedback (e.g., visualization, error messages) for each of the command user gave during the study. For commands that were incorrectly recognized, the researcher noted down the original command for analysis. For commands that were incorrectly interpreted or executed, the researcher marked the specific task and used the system log for further analysis. The command inference time was also recorded to evaluate the system responsiveness. To

focus on task-related input, we excluded user input that were not relevant to the task they were asked to perform. This could be due to misunderstanding of the task goal, or accidental misspoke commands. At the end, a questionnaire with Likert scale questions were used to evaluate the system usability.

For qualitative analysis, we audio-recorded semi-structured interviews with participants' permission, and conducted a thematic analysis [19] on the transcripts. The primary researcher generated the initial codes and themes, and collectively discussed and examined the results with the research team to reach consensus.

6 User Study Results

Below we present the study results. Unless otherwise noted, the findings are based on data from the test sessions.

6.1 RQ1: To what extent can participants complete a wide variety of tasks using HandProxy?

Participants successfully completed various tasks using HandProxy, with an 100% task completion rate on tasks conducted in the test sessions. Among the 781 commands participants issued across all test sessions, the system correctly executed 717 of them, with an overall accuracy of 91.8%. The median command execution accuracy per participant is 92.5%, with an IQR of 89.3% to 95.5%. On average, the system took 1.66 seconds to interpret the recognized speech command, with a standard deviation of 0.94 seconds.

After errors happened, **participants were able to recover from errors by using alternative commands to complete the task**. Multiple techniques were reported for error recovery, including rephrasing (13), repeating (3), and splitting into smaller steps (3). Across all commands in the test sessions, an average of 1.09 attempts (std 0.33) were needed for a command to succeed. Specifically, 85.5% of error commands were resolved with 1 more attempt, 12.7% took 2 more attempts, and 1.8% took 3 more attempts, showing that most errors were corrected with one alternative attempt.

6.2 RQ2: How effectively can the system interpret the diverse ways in which users issue commands?

HandProxy effectively handled users' diverse ways of phrasing commands, including less common ones. To identify unique commands per task, we lemmatized and removed stopwords from commands used by participants during test sessions. [Figure 7](#) shows the histogram and normalized entropy of unique commands participants used to complete study tasks. The normalized entropy quantifies the variation in the distribution of commands with a value from 0 (all attempts were the command) to 1 (each attempt corresponds to a unique command). Both the histogram and normalized entropy shows the diversity of commands participants used. Specifically, tasks were not dominated by just one command but were distributed across various commands, and the system was able to interpret most commands, including those used less frequently.

During practice sessions, participants generated an average of 32 unique commands per task. The system successfully interpreted command variations, including different verbs (e.g., {grab, pick up, fetch, hold} the peach), object descriptions (e.g., peach, pink fruit), spatial references (e.g., first fruit, second watermelon from the left), and sentence structures (e.g., can you use the minimize button, peach inside the basket). Moreover, the system demonstrated the ability to interpret and decompose high-level commands and generalize beyond literal meanings. For example, it interpreted “increase the brightness” as (i) grab the brightness knob and (ii) twist right. In the example of “fold the window,” the system was able to connect fold to minimize, and press the minimize button.

After practice sessions, as shown in [Figure 7](#), participants continued to use diverse commands rather than relying solely on the most straightforward ones, and the system effectively handled this variability. Given that participants were free to use their own words without predefined commands or formats, these results highlight HandProxy's capability to interpret flexible and varied input.

To better understand HandProxy's limitation, we analyzed the retrieved commands in the test sessions. In total, **the system failed to execute or incorrectly executed 64 commands**. Among them, 40 valid commands were categorized as invalid, and the system prompted users to try again and did not perform any action. Most were due to challenges for the LLM in identifying target objects ($n=26$), including synonyms (9), visual descriptions (6), object functionality (6), positional constraints (3), or other system errors (2). For example, participants described the power button as “the white and round button on the right side of the screen” (P1), the basket as “the brown object” (P16), or directly described the visual content within a widget, such as describing the window as “the dog photo” (P2). In other cases, participants referred to objects with synonyms or their expected functionality, such as describing the power button as “start on off button” (P10), or the heart button as “favorite button” (P16). While in many cases the system could infer beyond the object metadata, it struggled to account for the possible descriptions described above. This shows the need for a more comprehensive understanding of the objects and environment to support more diverse descriptive commands, which we will discuss in [section 7](#).

Apart from invalid commands with no action, the system incorrectly executed 24 valid commands. Among them, 3 were due to

speech recognition errors, while 18 resulted from misinterpreted hand interactions. These include ambiguous commands (6), incorrect command decomposition (5), incorrect gesture parsing (4), or performed on the wrong target objects (3). Some commands had multiple possible interpretations, where context could have clarified the intended meaning. For instance, when P6 said, “turn the volume slider to maximum,” the participant expected the slider to move to the top. However, the LLM interpreted the word “turn” literally and the system did a “twist right” gesture instead. Errors also occurred when participants combined multiple commands into one sentence. For example, P20 said, “pinch the resize button and pull it to the right”, expecting a continuous pinch-and-move gesture. Instead, the LLM parsed “pull” as a separate command, initiated a “grab” gesture that canceled the pinch and caused an unexpected action. The system identified the wrong object from some other commands. For example, P5 said “pick up the first watermelon on the right” to refer to the rightmost watermelon, but the system interpreted it as the first watermelon overall (left to right).

Nonetheless, participants were able to complete 100% of the tasks and recover from errors using alternative commands, with an average of 1.09 attempts (std 0.33) per command.

6.3 RQ3: What strategies, preferences, and methods do participants use for specifying intent?

Participants demonstrated diverse strategies and preferences while prompting the system. In practice sessions, the average command length was 5.25 words (std: 2.72, median: 5). In test sessions, the average was slightly lower at 4.73 words (std: 2.32, median: 4). For free exploration sessions – where users were not restricted to specific tasks – the average increased to 5.95 words (std: 3.32, median: 5). Participants who preferred more detailed commands found it easier to “specify what I want” (P2) or believed it would “help the system understand” their intent more accurately (P7). For example, to grab the peach, P15 used the command, “can you grab the peach and hold it in your hand?” In the interviews, 14 out of 20 participants favored shorter commands for their simplicity, clarity, and lower risk of errors. As P16 mentioned, “The longer it is, the more words I could say wrong and it could misinterpret.”

However, having shorter commands does not necessarily mean commands are always simple and low-level. In fact, participants used a mix of high-level and direct commands, such as “maximize the volume” (high-level) or “pinch the volume slider” followed by “pull up” (direct). For example, P9 mentioned that “I just wanted to tell the system what I want to do, and let the system figure out what gestures to do,” while others mentioned that using detailed control would be “more accurate” (P8), especially in precise tasks (P11) or those that need fine-grained controls (P6). Participants also reported the need to switch to a new mental model while using the system. This led some participants to use more descriptive ways of specifying their interaction intentions, such as “touch your index finger and thumb” to describe the pinch gesture (P13). As P17 said,

“A lot of the motions that you intuitively perform didn't come to me in words very easily. So my strategy was to describe exactly what the hand was doing until I'd sometimes realize, ‘Oh, that's how it should be [described]’ ”




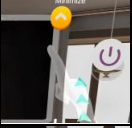

















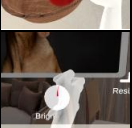








ID	Task	Task Image	Test Sessions				Example Commands	
			#	Rate	H_N	Distribution	Successful	Not Successful
1	Grab the {peach, apple, cube}		9	100%	0.83		- pick up the peach - grab the first fruit - grab the pink fruit	- grab the smallest fruit - grab the fifth fruit from the right - put your hand on the peach
			13	100%	0.88		- put your hand to the peach and hold it - go fetch the peach	- pick up the first fruit at the center of the table - grab the third item from the left
2	Press the {minimize, confirm, power} button		11	95%	0.85		- minimize the screen - click the minimize button - fold the image of the dog	- click on the up arrow - click on the upper right corner of the window
			16	83%	0.91		- press the yellow button that says minimize - can you use the minimize button	- point to the yellow arrow - I do not want to see the dog - push the minimize button
3	Pinch the {resize button, blue cube, volume slider}		8	91%	0.71		- pinch resize - do a pinch gesture at the corner of the resize	- resize - pinch the bottom right corner of the screen
			9	91%	0.8		- pinch on the resize - pinch on the resize button on the screen - do a pinch gesture at the resize button	- use the resize button - pick up resize - grab the resize button
4	Push the {confirm, like, power} button		12	91%	0.89		- push confirm - press the confirm button	- press the confirm button - palm that confirm button
			14	87%	0.81		- push the confirm button with your palm - slap the confirm button - move to the confirm button and push it	- hit the confirm button - press confirm button with palm - touch confirm button
5	Grab the watermelon {in the middle, on the left, on the right}		13	95%	0.92		- grab the second watermelon - grab the watermelon in the middle - grab the watermelon 2	- grab the second green object - hold the green ball in the middle of the table
			16	91%	0.96		- pick up the second watermelon - grab the watermelon, second from the left	- hold the green watermelon in the middle of the table - grab the watermelon to the right of the first watermelon
6	Move hand to the {left, up, forward}		13	100%	0.9		- move left - move hand leftward	- sweep your hand out to the left - move the hand across the screen
			13	80%	0.81		- move your hand from right to the left - push your hand to the left - slide the hand from right to left	- swipe the hand from right to left - move your hand across the room
7	Put the {apple, peach, blue cube} into the basket		14	100%	0.93		- drop the peach in the basket - grab the peach and put it in the basket	- peach in the basket
			16	95%	0.91		- move the peach in the basket - pick up peach; move hand up; move hand to the left; release - peach inside the basket	
8	Rotate the brightness knob {clockwise, counterclockwise}		17	95%	0.96		- increase the brightness - turn the brightness knob to the right	- turn up the brightness button
			18	100%	0.99		- twist brightness to the right - make the photo brighter using the brightness button - screw the brightness button rightward	
9	Move {apple, peach, cube} to the left following traffic light signal		19	100%	0.99		- move the apple to the left; stop; move left - grab the apple; go to the left; stop; go	- move the apple to the left when the light is green - pick up the apple and move left when the light is green and stop moving when the light is yellow or red
			22	91%	1		- hold the apple; move left to the screen; stop; go - pick the apple and move your hand in the left direction; stop; move left - pick up the apple and move it to the left; stop; go	- move the apple outside the screen - move apple to the left according to the traffic light - move left when the circle on the top of the screen is green
10	Press and hold the {power, like, confirm} button		14	100%	0.94		- point the power button and keep your finger there - keep pointing to the power button	- point and push the power button
			13	100%	0.84		- long press power button - hold the power button- apply pressure to the power button	- push the power button and keep your hand on it - point and press the power button

Figure 7: Command diversity in test sessions: Each task was repeated on three objects — one for practice and two for test sessions (top row: second object, bottom row: third object). We report the number of unique command phrasings (#), execution accuracy (Rate), normalized entropy (H_N), and a histogram showing the distribution of unique phrasings, sorted by frequency. We color-coded the successful attempts as blue and unsuccessful attempts as red to illustrate the distribution of correct/incorrect commands. Finally, we provide examples of successful and unsuccessful participant commands.

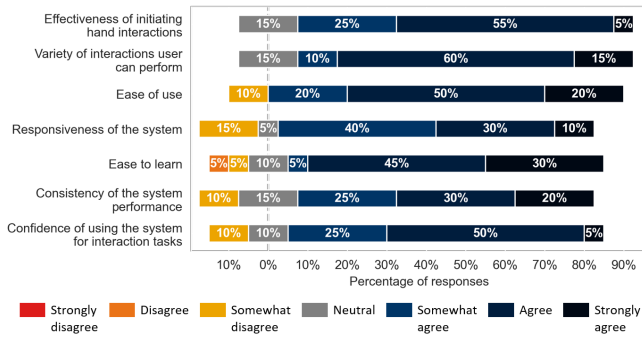


Figure 8: Final Likert scale responses of system usability.

6.4 RQ4: How do participants experience and perceive HandProxy?

We evaluated system usability using 7-point Likert scale questions, covering system effectiveness, interaction variety, ease of use, responsiveness, learnability, consistency, and user confidence. The questions are based on the System Usability Scale [20], with modifications to better align with the specific tasks in this user study. The Likert scale questions were administered three times at different stages to identify potential learning effects. However, no significant differences were found between them. Therefore, we report the results from the final administration that were conducted after all tasks were completed, as they provide the most comprehensive reflection of participants' overall experience with the system.

Overall, participants found HandProxy effective for hand interactions (avg. 5.5, std. 0.82), can support various interaction gestures (average 5.75, std. 0.91), easy to use (average 5.7, std. 1.12), and participants were generally confident in using it (average 5.3, std. 1.08). Regarding system responsiveness, participants gave an average of 5.15 (std. 1.18). While participants appreciated the fast timing controls (e.g., stop, continue, undo) (e.g., P6), overall responsiveness across different types of commands could be improved for smoother interaction (P2, P8). HandProxy was also seen as intuitive and easy to learn (avg. 5.7, std. 1.41), especially given that the study was intentionally designed to have only a very brief on-boarding session. As P3 mentioned, "a few minutes of play and you should be ready." However, P15 highlighted challenges for users unfamiliar with AR/VR, often relying on long, descriptive commands and struggling with terminology. Both P15 and P11 suggested that a tutorial with example commands, words, and objects would make the system more intuitive for beginners.

Additionally, participants valued HandProxy's ability to handle high-level commands (P1, P2, P8, P11, P19); robustness to certain ambiguity in the command (P5, P10, P11); consistency on the same task (P7, P15); its memory capability (P19); and its ability to take continuous speech input and perform actions on the go (P7). For example, P1 mentioned that HandProxy is able to directly execute her high-level commands step by step, "even though I haven't told the system how to perform the actual task."

Interestingly, we found that **participants had different perceptions of the virtual proxy hand – either as an interaction tool, an agent, or a part of their body**. These not only impacted the way they used it, but also their expectations about the system's

capabilities. Participants who perceived HandProxy as a tool treated the virtual hand more like a cursor, and cared more about what it could achieve than how realistic the interaction was, and in some cases, suggested that it should go beyond what a real hand can do. As P11 mentioned,

"I don't know if the system can make the hand more abstract. For example, it could hold many things at a time. It may not need to be limited to the physical world, allowing for actions beyond what we can do in real life."

This perception of HandProxy as a tool also influenced how participants talked to the system. For example, P7 used simple, direct, "non-human speech" to complete tasks, "just like other voice controls that's relatively old." This perception may have led some participants to underestimate the system's ability to handle high-level, complex input. As P14 said, "I thought it only do steps, but later found it does whole tasks and I will do that."

However, participants who perceived the system as an agent would prefer to treat it as an assistant or "you," expecting it to understand high-level commands, or even answer questions related to the environment. For example, participants mentioned that talking to this system is similar to "talk[ing] normally and intuitively like you would to a person" (P2). When there is confusion, participants may expect to get answers from the system. For example, P18 hoped the system could help him identify objects when he forgot their names: "for example, I could say, What's that green-striped object? and it could respond, It's a watermelon."

Additionally, some participants treated the virtual hand as an extension of their body that just "behaves like my arm" (P12), which could enhance immersion in the virtual environment. P13 said,

"Essentially, you want to immerse your hand into this [...] virtual environment. It's about bridging the gap between you and the virtual space, allowing direct interaction with what's inside, like in Minecraft or other virtual worlds. You'd want to 'put your hand in' and interact with objects or perform tasks as if you were physically present in that environment."

6.5 RQ5: What other expectations do participants have for the system?

In the free exploration session, many participants tried commands requiring multimodal understanding of the environment, context, and how different objects are related to each other. For example, P18 wanted to say "like the dog" to press the heart button on top of the image viewer window. This expectation of environment understanding also extends to how the physics work in the virtual environment, such as "throw the apple to the dog" and "catch the apple right before it hits the table." Commands based on visual descriptions were also used. As P19 mentioned,

"For example, if I forgot the name of an object, like a watermelon, and describe it as a 'green-striped round object,' I'd expect it to recognize the object. In real life, there are moments when you can't recall names, so being able to describe objects and have the system understand would be helpful."

Participants also tried additional gestures that the system did not yet directly support, such as roll, throw, flip, wipe, and detailed finger controls. While there was no direct match to these gestures,

HandProxy still tried to utilize existing gestures to reproduce the command to its best. For example, the system performed a grab and rotate to “flip a basket.” In addition, participants also suggested other hand controls that could be useful, including bimanual control (P1, P2, P13), supporting more precise, affordance-dependent object interaction (P2, e.g., grab the basket handle/side), and detailed individual finger controls (P5). Additionally, participants would like the system to understand object states and conditions in the environment. For example, P20 tried “if the photo has been resized, press the confirm button.” This also includes monitoring the state of the object while being manipulated by the virtual hand. For example, P13 mentioned that, “There was one time when it was trying to move the apple but didn’t realize it no longer had the apple in its hand,” if the system was aware of the hand state, it could automatically correct this error. These require understanding and monitoring the object state to relieve users from “monitoring conditions constantly.” (P19) Participants also emphasized the importance of additional feedback to help users identify and recover from errors. For example, P6 recommended underlining the parts in user’s command that are confusing or uninterpretable, enabling users to rephrase their input more effectively in subsequent attempts. P1 suggested having the system predict and suggest possible next moves after the command, such as showing all afforded hand interactions once the user grabbed an object. Additionally, participants noted that improving the speech recognition system and reducing overall latency could enhance the system’s usability. Current limitations and possible extensions on HandProxy are further discussed in [section 7](#).

7 Discussion and Future Work

Here, we expand on our key findings and discuss their implications for future work, including handling ambiguity, system transparency, supporting bimanual interaction, and creating a unified accessibility API for hand interaction.

7.1 Handling Ambiguity in Interaction Commands

Ambiguity is inherently a part of the natural language, as certain commands could be interpreted in different ways. Addressing this challenge requires understanding the user’s true intentions. Based on observations from the user study, two key questions arise: (i) what additional information can be leveraged to resolve ambiguity, and (ii) how should we balance the system’s ability to disambiguate and requesting clarifications from users.

During the user study, the commands that were considered “ambiguous” by the system were often due to insufficient understanding of the environment and the target objects, including visual features, complex spatial referencing, and object affordance. The current disambiguation design is mostly focusing on disambiguating duplicate objects. However, as discovered in the user study, additional ambiguity could come from the use of synonyms, or commands that have multiple interpretations. In cases of ambiguity, the system sometimes took initiatives and filled in the necessary information. While this worked in some cases, it inevitably caused unexpected behavior (e.g., interpreting “turn the volume slider to maximum” as rotate volume knob to the right, rather than moving up). This highlights a design challenge that even with additional information

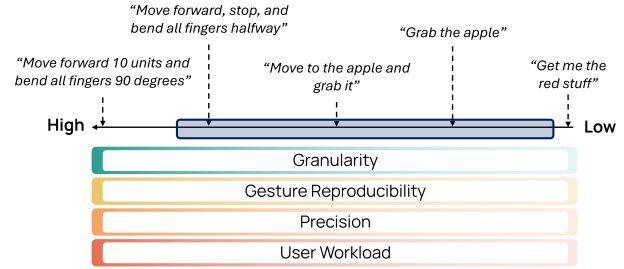


Figure 9: Spectrum of command granularity, with an estimated preferred range highlighted on the spectrum.

about the environment, it is important to determine what can and should be automatically inferred by the system, and what should be prompted to users for clarification. An avenue for future work is to investigate how the system should balance automated inference and user clarification to consider both accuracy and user effort.

7.2 Supporting Multiple Levels of Interaction Control

Hand interaction commands can vary in granularity along the spectrum shown in [Figure 9](#), with trade-offs in gesture reproducibility, ambiguity, user workload, and system capability. At the high-granularity end of the spectrum, users have full, precise control of the hand. This allows users to reproduce almost any hand gesture and even fine-tune or customize gestures. While such commands are less ambiguous, they impose a higher cognitive and physical workload on users due to the detailed input required. Conversely, at the low-granularity end of the spectrum, users can issue abstract, high-level interaction goals, leaving the system to determine how and what to do to achieve the goal. While this reduces user effort and simplifies command input, it comes with the trade-offs of reduced precise control, increased ambiguity, and a greater demand on the system’s interpretation capabilities.

To design effective and efficient speech to hand controls, it is important to define an appropriate range of supported commands that balances granularity and usability. We observed and estimated a preferred range for hand controls from our study, illustrated in [Figure 9](#). The preferred range skews toward the low-granularity end of the spectrum, with spacings on both left and right. The left end of the spectrum, although having more hand control possibilities, was not included because of the control complexity. Although participants preferred high-level commands, they may require significant system intelligence, and would likely introduce unnecessary ambiguity. These initial findings provide possible considerations for designing speech interfaces for hand interaction controls. However, further studies are needed to refine our understanding of the optimal level of command granularity and its impact on usability.

7.3 Improving Transparency and Gesture Discoverability

During the user study, participants often struggled to identify the specific part of their commands caused issues when errors occurred. We believe the feedback system could be improved by

providing more details about how the system interprets commands, and should highlight the ambiguous parts and how the system interpreted them. These could be achieved through a better-designed text feedback of the recognized command, such as to color-code the words in the command by the level of ambiguity and highlight the parts that could not be interpreted. Without overwhelm users with excessive feedback, such improvements could enhance the transparency of the system, help users effectively refine their inputs and improve their overall experience.

In addition to addressing errors, the interactions' discoverability remains an area for improvement. While flexible speech interfaces alleviate the challenge of knowing exactly what to say, participants may still struggle to identify the possible affordances for a given object. Although HandProxy is able to infer expected gestures from high-level commands to some extent, it could be further improved by proactively showing possible interactions. For example, P3 suggested showing a list of possible next steps after each interaction, such as options like "throw," "squeeze," or "drop" after an apple is grabbed. This could not only help users discover additional affordances but also accelerate workflows by suggesting the most relevant next steps based on the interaction history.

7.4 Supporting Bimanual Interactions

Bimanual interactions offer greater possibilities than unimanual (i.e., one-handed) interactions. Immersive apps such as Paper Birds² and Cubism³ utilize both hands either collaboratively (e.g., performing a joint action) or separately (e.g., one hand for object manipulation, the other for view control). While we demonstrated the feasibility of speech-controlled one-handed interactions, the system could be extended to support bimanual interactions — either controlling both hands or a single proxy hand collaborating with the user's real hand.

Future work could expand the design space for bimanual interactions along two key dimensions: (i) the type of bimanual interactions, and (ii) the level of proxy controls. For interaction types, Yamagami et al. [86] categorized bimanual interactions into symmetric in-phase (e.g., jump-roping), symmetric out-of-phase (e.g., climb a ladder), asymmetric coordinated (e.g., swing a golf club), and asymmetric uncoordinated (e.g., use two swords at the same time). Regarding the level of proxy controls, the system could be designed to perform bimanual interactions directly through two proxy hands, or employ one proxy hand that monitors, interprets, and collaborates with the user's one-handed input. This expanded design space could better support bimanual interactions.

7.5 Limitation on Applicable Use Cases

While the proposed approach is designed to generalize across a wide range of interaction scenarios, it may not always offer substantial advantages over alternative methods. For more direct and less hand-dependent interactions — such as basic system operations (e.g., power on/off) or standard UI controls (e.g., closing windows) — a traditional speech interface with direct command mapping may be more effective. Interactions requiring high precision or complexity, such as fine-grained rotations in training or simulation-based

XR applications, may pose challenges when relying solely on the proposed speech interface. Furthermore, although our approach leverages commonly used gestures across various immersive applications, the current implementation does not readily accommodate applications that heavily depend on customized, non-standard hand gestures. For example, the spell-casting game Drakheir Hands of Wizard⁴ requires specialized gesture inputs that are not easily replicated within the existing system. These observations suggest that further enhancements are needed to support a broader range of use cases, particularly those involving diverse gesture types, varying levels of precision, and additional control requirements.

As an initial exploration of the proxied interaction paradigm for speech interfaces, HandProxy shows the potential of using virtual hands to broaden interaction possibilities. Future work could address current limitations through the integration of multimodal input strategies that adaptively optimize interaction based on task type and complexity. Additional improvements might include support for runtime gesture recording to enable customizable gesture macros, as well as integration with system-level APIs to facilitate cross-application control.

7.6 Towards an Accessible Interface for Hand Interactions

A key motivation of this work is to find an expressive, flexible alternative to hand interactions in cases of situational impairments, ability mismatches, or user's varied preferences. While this work demonstrated how speech interfaces can be enhanced to achieve this goal, it brings up a broader question: can we define a unified control interface for possible hand interactions in the virtual environment — just like defining an API for 2D cursor controls, so that the interactions can be mapped to a broader range of input modality setups, potentially accommodating a wide range of user's abilities and preferences?

To achieve this, it is important to create a comprehensive, well-defined set of vocabularies for hand interactions in the virtual environment. Then, the direct mapping between the hand input and an arbitrary input modality could be simplified to the mapping to this shared vocabulary. This approach could enable flexible and extensible mapping of hand input to other input setups, including multimodal configurations, that could be used to provide a more accessible way of interacting with the virtual environment.

8 Conclusion

We presented HandProxy, a system that enables users to control a virtual proxy hand using natural speech commands, allowing it to perform various hand interactions on the user's behalf. To achieve this, we defined a set of hand control primitives and demonstrated how different hand interactions can be composed by combining these primitives. Building on this structure, we implemented HandProxy as a real-time system that supports the continuous streaming and execution of user commands with varying levels of granularity. Through a user study with 20 participants, we demonstrated that HandProxy effectively enables users to complete a wide range of tasks typically designed for direct hand interactions, and

²<https://www.3dar.com/p/paper-birds>

³<https://www.cubism-vr.com/>

⁴<https://www.meta.com/experiences/drakheir-hands-of-wizard>

showed that HandProxy is able to interpret diverse command variations. Additionally, we explored user strategies, preferences, and expectations regarding speech-driven hand interactions. Finally, we reflected on key findings from the study and discussed their implications for future developments, including resolving ambiguity in user commands, supporting varying levels of interaction control, enhancing system transparency and gesture discoverability, supporting bimanual interactions, and directions towards an accessible interface in virtual environment. This work demonstrates the potential of speech interfaces, augmented by interaction proxies, to expand their capabilities and facilitate more expressive interactions. Our findings highlight new possibilities for initiating expressive interactions through speech interfaces and point to future directions for enhancing usability, adaptability, and intelligent interaction proxies in virtual environments.

A Example Gesture Metadata File

```
{
  "name": "cut",
  "data_format": "unified",
  "data_source": "leap_motion",
  "num_hands": 1,
  "right_hand_data_file": "cut.txt",
  "left_hand_data_file": null,
  "is_hold_at_peak": false,
  "is_static": false,
  "interacting_frame": 81,
  "interacting_joint": ["pinky_mcp"],
  "segments": [
    {
      "name": "preparation",
      "start_frame": 0,
      "end_frame": 58
    },
    {
      "name": "stroke",
      "start_frame": 58,
      "end_frame": 123
    },
    {
      "name": "retraction",
      "start_frame": 123,
      "end_frame": 200
    }
  ]
}
```

References

- [1] 2024. Browse Amazing Hand Tracking Apps VR Games on Meta Quest | Meta Store. <https://www.meta.com/experiences/section/344493996865090/>
- [2] 2024. Control Apple Vision Pro by making sounds. <https://support.apple.com/guide/apple-vision-pro/perform-actions-with-sounds-tan319fb0f99/visionos>
- [3] 2024. Getting started with Hand and Body Tracking on Meta Quest headsets. <https://www.meta.com/help/quest/articles/headsets-and-accessories/controllers-and-hand-tracking/hand-tracking/>
- [4] 2024. HoloLens 2 gestures for authoring and navigating in Dynamics 365 Guides - Dynamics 365 Mixed Reality. <https://learn.microsoft.com/en-us/dynamics365/mixed-reality/guides/authoring-gestures-hl2>
- [5] 2024. Use and manage Meta AI assistant on Meta Quest. <https://www.meta.com/help/quest/articles/in-vr-experiences/oculus-features/use-manage-meta-ai/>
- [6] 2024. Use gestures with Apple Vision Pro. <https://support.apple.com/en-us/117741>
- [7] 2024. Use Voice Control to interact with Apple Vision Pro. <https://support.apple.com/guide/apple-vision-pro/perform-actions-with-your-voice-tan14d179ad1/visionos>
- [8] 2024. Voice input. <https://learn.microsoft.com/en-us/windows/mixed-reality/design/voice-input>
- [9] Setareh Aghel Manesh, Tianyi Zhang, Yuki Onishi, Kotaro Hara, Scott Bateman, Jiannan Li, and Anthony Tang. 2024. How people prompt generative AI to create interactive VR scenes. In *Designing Interactive Systems Conference*. ACM, New York, NY, USA.
- [10] A.M.R. Agur, M.J. Lee, and J.C.B. Grant. 1999. *Grant's Atlas of Anatomy*. Lippincott Williams & Wilkins. <https://books.google.com/books?id=8RlqAAAAAMAAJ>
- [11] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. 2022. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. *arXiv:2204.01691* [cs.RO] <https://arxiv.org/abs/2204.01691>
- [12] Apple. 2024. Use Voice Control commands to interact with iPhone. <https://support.apple.com/guide/iphone/use-voice-control-iph2c21a3c88/ios>
- [13] Apple. 2024. Use Voice Control on your Mac. <https://support.apple.com/en-us/102225>
- [14] Rahul Arora, Rubaiat Habib Kazi, Danny M. Kaufman, Wilmot Li, and Karan Singh. 2019. MagicalHands: Mid-Air Hand Gestures for Animating in VR. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) (*UIST '19*). Association for Computing Machinery, New York, NY, USA, 463–477. <https://doi.org/10.1145/3332165.3347942>
- [15] Farhan Aslam and Richard Zhao. 2024. Voice-Augmented Virtual Reality Interface for Serious Games. In *2024 IEEE Conference on Games (CoG)*. 1–8. <https://doi.org/10.1109/CoG60054.2024.10645616>
- [16] Huidong Bai, Gun A. Lee, Mukundan Ramakrishnan, and Mark Billinghurst. 2014. 3D gesture interaction for handheld augmented reality. In *SIGGRAPH Asia 2014 Mobile Graphics and Interactive Applications* (Shenzhen, China) (*SA '14*). Association for Computing Machinery, New York, NY, USA, Article 7, 6 pages. <https://doi.org/10.1145/2669062.2669073>
- [17] Mark Billinghurst, Hirokazu Kato, and Ivan Poupyrev. 2001. The MagicBook: a transitional AR interface. *Computers & Graphics* 25, 5 (2001), 745–753. [https://doi.org/10.1016/S0097-8493\(01\)00117-0](https://doi.org/10.1016/S0097-8493(01)00117-0) Mixed realities - beyond conventions.
- [18] Richard A. Bolt. 1980. "Put-that-there": Voice and gesture at the graphics interface. In *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques* (Seattle, Washington, USA) (*SIGGRAPH '80*). Association for Computing Machinery, New York, NY, USA, 262–270. <https://doi.org/10.1145/800250.807503>
- [19] Virginia Braun and Victoria Clarke. 2012. Thematic analysis. In *APA handbook of research methods in psychology, Vol 2: Research designs: Quantitative, qualitative, neuropsychological, and biological*. American Psychological Association, Washington, 57–71.
- [20] John Brooke. 1996. SUS-A quick and dirty usability scale (in "Usability Evaluation in Industry", PW Jordan, B Thomas, I McLelland, BA Weerdmeester (eds)). 194 (1996), 189–194.
- [21] Micael Carreira, Karine Lan Ting, Petra Csobanka, and Daniel Gonçalves. 2017. Evaluation of in-air hand gestures interaction for older people. *Univers. Access Inf. Soc.* 16, 3 (aug 2017), 561–580. <https://doi.org/10.1007/s10209-016-0483-y>
- [22] Junuk Cha, Jiyeon Kim, Jae Shin Yoon, and Seungryul Baek. 2024. Text2HOI: Text-guided 3D Motion Generation for Hand-Object Interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1577–1585.
- [23] Neil Chulpongatorn, Wesley Willett, and Ryo Suzuki. 2023. HoloTouch: Interacting with Mixed Reality Visualizations Through Smartphone Proxies. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) (*CHI EA '23*). Association for Computing Machinery, New York, NY, USA, Article 156, 8 pages. <https://doi.org/10.1145/3544549.3585738>
- [24] Andrea Coppari, Silvia Proia, Andrea Ruio, Filippo Favali, Lorenzo Sabattini, Cristian Secchi, Valeria Villani, Marco Piazzola, and Luca Capra. 2025. A Large Language Model-Based Motion Planning for Human-Robot Interaction: An Experimental Case Study. In *Human-Friendly Robotics 2024*, Antonio Paolillo, Alessandro Giusti, and Gabriele Abbate (Eds.). Springer Nature Switzerland, Cham, 99–113.
- [25] Fernanda De La Torre, Cathy Mengying Fang, Han Huang, Andrzej Banburski-Fahy, Judith Amores Fernandez, and Jaron Lanier. 2024. LLMR: Real-time prompting of interactive worlds using large language models. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA.
- [26] William Delamare, Chaklam Silpasuwanchai, Sayan Sarcar, Toshiaki Shiraki, and Xiangshi Ren. 2019. On Gesture Combination: An Exploration of a Solution to Augment Gesture Interaction. In *Proceedings of the 2019 ACM International Conference on Interactive Surfaces and Spaces* (Daejeon, Republic of Korea) (*ISS '19*). Association for Computing Machinery, New York, NY, USA, 135–146. <https://doi.org/10.1145/3343055.3359706>
- [27] Mustafa Doga Dogan, Eric J Gonzalez, Karan Ahuja, Ruofei Du, Andrea Colaço, Johnny Lee, Mar Gonzalez-Franco, and David Kim. 2024. Augmented Object Intelligence with XR-Objects. In *Proceedings of the 37th Annual ACM Symposium*

- on *User Interface Software and Technology* (Pittsburgh, PA, USA) (UIST '24). Association for Computing Machinery, New York, NY, USA, Article 19, 15 pages. <https://doi.org/10.1145/3654777.3676379>
- [28] Cathy Mengying Fang, Krzysztof Zieliński, Pattie Maes, Joe Paradiso, Bruce Blumberg, and Mikkel Baun Kjergaard. 2024. Enabling Waypoint Generation for Collaborative Robots using LLMs and Mixed Reality. arXiv:2403.09308 [cs.HC] <https://arxiv.org/abs/2403.09308>
- [29] George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. 1987. The vocabulary problem in human-system communication. *Commun. ACM* 30, 11 (1987), 964–971.
- [30] Guillermo Garcia-Hernando, Shanxin Yuan, Seungryul Baek, and Tae-Kyun Kim. 2018. First-Person Hand Action Benchmark with RGB-D Videos and 3D Hand Pose Annotations. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*.
- [31] Yalda Ghasemi and Heejin Jeong. 2022. Using Gaze-based Interaction to Alleviate Situational Mobility Impairment in Extended Reality. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 66, 1 (2022), 435–439. <https://doi.org/10.1177/1071181322661224> arXiv:https://doi.org/10.1177/1071181322661224
- [32] Google. 2024. Principles for improving app accessibility | App quality. <https://developer.android.com/guide/topics/ui/accessibility/principles#label-elements>
- [33] Google. 2024. Speech-to-Text request construction | Cloud Speech-to-text Documentation | Google Cloud. <https://cloud.google.com/speech-to-text/docs/speech-to-text-requests>
- [34] Devamardeep Hayatpur, Seongkook Heo, Haijun Xia, Wolfgang Stuerzlinger, and Daniel Wigdor. 2019. Plane, ray, and point: Enabling precise spatial manipulations with shape constraints. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA.
- [35] Daniel Hepperle, Yannick Weiß, Andreas Siess, and Matthias Wölfel. 2019. 2D, 3D or speech? A case study on which user interface is preferable for what kind of object interaction in immersive virtual reality. *Comput. Graph.* 82 (Aug. 2019), 321–331.
- [36] Anuruddha Hettiarachchi and Daniel Wigdor. 2016. Annexing Reality: Enabling Opportunistic Use of Everyday Objects as Tangible Proxies in Augmented Reality. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) (CHI '16). Association for Computing Machinery, New York, NY, USA, 1957–1967. <https://doi.org/10.1145/2858036.2858134>
- [37] Juan David Hincapié-Ramos, Xiang Guo, Paymahn Moghadasian, and Pourang Irani. 2014. Consumed endurance: a metric to quantify arm fatigue of mid-air interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) (CHI '14). Association for Computing Machinery, New York, NY, USA, 1063–1072. <https://doi.org/10.1145/2556288.2557130>
- [38] Jan Hombeck, Henrik Voigt, Timo Heggemann, Rabi R. Datta, and Kai Lawonn. 2023. Tell Me Where To Go: Voice-Controlled Hands-Free Locomotion for Virtual Reality Systems. In *2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*. 123–134. <https://doi.org/10.1109/VR55154.2023.00028>
- [39] Masoumehsadat Hosseini, Tjado Ihmels, Ziqian Chen, Marion Koelle, Heiko Müller, and Susanne Boll. 2023. Towards a Consensus Gesture Set: A Survey of Mid-Air Gestures in HCI for Maximised Agreement Across Domains. 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 311, 24 pages. <https://doi.org/10.1145/3544548.3581420>
- [40] Brandi House, Jonathan Malkin, and Jeff Bilmes. 2009. The VoiceBot: a voice controlled robot arm. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Boston, MA, USA) (CHI '09). Association for Computing Machinery, New York, NY, USA, 183–192. <https://doi.org/10.1145/1518701.1518731>
- [41] Takeo Igarashi and John F. Hughes. 2001. Voice as sound: using non-verbal voice input for interactive control. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology* (Orlando, Florida) (UIST '01). Association for Computing Machinery, New York, NY, USA, 155–156. <https://doi.org/10.1145/502348.502372>
- [42] Rahul Jain, Jingyu Shi, Runlin Duan, Zhengzhe Zhu, Xun Qian, and Karthik Ramani. 2023. Ubi-TOUCH: Ubiquitous Tangible Object Utilization through Consistent Hand-object interaction in Augmented Reality. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology* (San Francisco, CA, USA) (UIST '23). Association for Computing Machinery, New York, NY, USA, Article 12, 18 pages. <https://doi.org/10.1145/3586183.3606793>
- [43] Sujin Jang, Wolfgang Stuerzlinger, Satyajit Ambike, and Karthik Ramani. 2017. Modeling Cumulative Arm Fatigue in Mid-Air Interaction based on Perceived Exertion and Kinetics of Arm Motion. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) (CHI '17). Association for Computing Machinery, New York, NY, USA, 3328–3339. <https://doi.org/10.1145/3025453.3025523>
- [44] Yu Jiang, Zhipeng Li, Mufei He, David Lindlbauer, and Yukang Yan. 2023. HandAvatar: Embodying Non-Humanoid Virtual Avatars through Hands. 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 309, 17 pages. <https://doi.org/10.1145/3544548.3581027>
- [45] Maria Karam and m.c Schraefel. 2005. A Taxonomy of Gestures in Human Computer Interactions. (01 2005).
- [46] Mohamed Kari and Christian Holz. 2023. HandyCast: Phone-based Bimanual Input for Virtual Reality in Mobile and Space-Constrained Settings via Pose-and-Touch Transfer. 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 528, 15 pages. <https://doi.org/10.1145/3544548.3580677>
- [47] Toby Jia-Jun Li, Amos Azaria, and Brad A. Myers. 2017. SUGILITE: Creating Multimodal Smartphone Automation by Demonstration. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) (CHI '17). Association for Computing Machinery, New York, NY, USA, 6038–6049. <https://doi.org/10.1145/3025453.3025483>
- [48] Chen Liang, Yasha Irvantchi, Thomas Krolkowski, Ruijie Geng, Alanson P. Sample, and Anhong Guo. 2023. BrushLens: Hardware Interaction Proxies for Accessible Touchscreen Interface Actuation. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology* (San Francisco, CA, USA) (UIST '23). Association for Computing Machinery, New York, NY, USA, Article 36, 17 pages. <https://doi.org/10.1145/3586183.3606730>
- [49] Frieder Loch. 2012. Hierarchical gestures: ggestural shortcuts for touchscreen devices. <https://essay.utwente.nl/61931/>
- [50] Tao Lu, Hongxiao Zheng, Tianying Zhang, Xuhai “Orson” Xu, and Anhong Guo. 2024. InteractOut: Leveraging Interaction Proxies as Input Manipulation Strategies for Reducing Smartphone Overuse. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (CHI '24). Association for Computing Machinery, New York, NY, USA, Article 245, 19 pages. <https://doi.org/10.1145/3613904.3642317>
- [51] Scott Mcglashan and Tomas Axling. 1996. A Speech Interface to Virtual Environments. (11 1996).
- [52] David McNeill. 1992. Hand and mind. *Advances in Visual Semiotics* 351 (1992).
- [53] Pedro Monteiro, Guilherme Goncalves, Hugo Coelho, Miguel Melo, and Maximino Bessa. 2021. Hands-free interaction in immersive virtual reality: A systematic review. *IEEE Trans. Vis. Comput. Graph.* 27, 5 (May 2021), 2702–2713.
- [54] Martez Mott, John Tang, Shaun Kane, Edward Cutrell, and Meredith Ringel Morris. 2020. “I just went into it assuming that I wouldn’t be able to have the full experience”: Understanding the Accessibility of Virtual Reality for People with Limited Mobility. In *Proceedings of the 22nd International ACM SIGACCESS Conference on Computers and Accessibility* (Virtual Event, Greece) (ASSETS '20). Association for Computing Machinery, New York, NY, USA, Article 43, 13 pages. <https://doi.org/10.1145/3373625.3416998>
- [55] Mozilla. 2024. Introduction to the DOM. https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction
- [56] Christine Murad, Heloisa Candello, and Cosmin Munteanu. 2023. What’s The Talk on VUI Guidelines? A Meta-Analysis of Guidelines for Voice User Interface Design. In *Proceedings of the 5th International Conference on Conversational User Interfaces* (CUI '23). Association for Computing Machinery, New York, NY, USA.
- [57] Robin R Murphy. 2019. *Introduction to AI robotics*. MIT press.
- [58] OpenAI. 2024. Models - OpenAI API. <https://platform.openai.com/docs/models#gpt-4o>
- [59] Akhil Padmanabha, Jessie Yuan, Janavi Gupta, Zulekha Karachiwalla, Carmel Majidi, Henny Admoni, and Zackory Erickson. 2024. VoicePilot: Harnessing LLMs as Speech Interfaces for Physically Assistive Robots. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA, 1–18.
- [60] Siyou Pei, Alexander Chen, Jaewook Lee, and Yang Zhang. 2022. Hand Interfaces: Using Hands to Imitate Objects in AR/VR for Expressive Interactions. 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 429, 16 pages. <https://doi.org/10.1145/3491102.3501898>
- [61] Tran Pham, Jo Vermeulen, Anthony Tang, and Lindsay MacDonald Vermeulen. 2018. Scale Impacts Elicited Gestures for Manipulating Holograms: Implications for AR Gesture Design. In *Proceedings of the 2018 Designing Interactive Systems Conference* (Hong Kong, China) (DIS '18). Association for Computing Machinery, New York, NY, USA, 227–240. <https://doi.org/10.1145/3196709.3196719>
- [62] Thammathip Piumsomboon, David Altimira, Hyungon Kim, Adrian Clark, Gun Lee, and Mark Billinghurst. 2014. Grasp-Shell vs gesture-speech: A comparison of direct and indirect natural interaction techniques in augmented reality. In *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 73–82. <https://doi.org/10.1109/ISMAR.2014.6948411>
- [63] Thammathip Piumsomboon, Adrian Clark, Mark Billinghurst, and Andy Cockburn. 2013. User-defined gestures for augmented reality. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems* (Paris, France) (CHI EA '13). Association for Computing Machinery, New York, NY, USA, 955–960. <https://doi.org/10.1145/2468356.2468527>
- [64] Ivan Poupyrev, Mark Billinghurst, Suzanne Weghorst, and Tadao Ichikawa. 1996. The go-go interaction technique: non-linear mapping for direct manipulation in VR. In *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology* (Seattle, Washington, USA) (UIST '96). Association for Computing

- Machinery, New York, NY, USA, 79–80. <https://doi.org/10.1145/237091.237102>
- [65] Ivan Poupyrev and Tadao Ichikawa. 1999. Manipulating objects in virtual worlds: Categorization and empirical evaluation of interaction techniques. *Journal of Visual Languages & Computing* 10, 1 (1999), 19–35.
- [66] Jing Qian, Jiaju Ma, Xiangyu Li, Benjamin Attal, Haoming Lai, James Tompkin, John F. Hughes, and Jeff Huang. 2019. Portal-ble: Intuitive Free-hand Manipulation in Unbounded Smartphone-based Augmented Reality. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) (UIST '19). Association for Computing Machinery, New York, NY, USA, 133–145. <https://doi.org/10.1145/3332165.3347904>
- [67] Yanayir Rifai, Ahmad Ataka, Agus Bejo, and Yusuf Kurnia Badriawan. 2024. Upper Limb Rehabilitation Robot Control based on Large Language Model. In *2024 International Conference on Computer, Control, Informatics and its Applications (IC3INA)*. 422–427. <https://doi.org/10.1109/IC3INA64086.2024.10732179>
- [68] Jonas Schjerlund, Kasper Hornbæk, and Joanna Bergström. 2021. Ninja Hands: Using Many Hands to Improve Target Selection in VR. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) (CHI '21). Association for Computing Machinery, New York, NY, USA, Article 130, 14 pages. <https://doi.org/10.1145/3411764.3445759>
- [69] Rajeev Sharma, Thomas S. Huang, and Vladimir I. Pavlovic. 1996. A Multimodal framework for Interacting with Virtual Environments. Springer US, Boston, MA, 53–71. https://doi.org/10.1007/978-1-4613-1447-9_5
- [70] Rajinder S. Sodhi, Brett R. Jones, David Forsyth, Brian P. Bailey, and Giuliano Maciocco. 2013. BeThere: 3D mobile collaboration with spatial input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) (CHI '13). Association for Computing Machinery, New York, NY, USA, 179–188. <https://doi.org/10.1145/2470654.2470679>
- [71] Misha Sra, Xuhai Xu, and Pattie Maes. 2018. BreathVR: Leveraging Breathing as a Directly Controlled Interface for Virtual Reality Games. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) (CHI '18). Association for Computing Machinery, New York, NY, USA, 1–12. <https://doi.org/10.1145/3173574.3173914>
- [72] Péter Telkes, Alexandre Angleraud, and Roel Pieters. 2024. Instructing Hierarchical Tasks to Robots by Verbal Commands. In *2024 IEEE/SICE International Symposium on System Integration (SII)*. 1139–1145. <https://doi.org/10.1109/SII58957.2024.10417491>
- [73] Jingze Tian, Yingna Wang, Keyue Yu, Liyi Xu, Junan Xie, Franklin Mingzhe Li, Yafeng Niu, and Mingming Fan. 2024. Designing Upper-Body Gesture Interaction with and for People with Spinal Muscular Atrophy in VR. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (CHI '24, Article 60). Association for Computing Machinery, New York, NY, USA, 1–19.
- [74] Wen-Jie Tseng, Samuel Huron, Eric Lecolinet, and Jan Gugenheimer. 2023. FingerMapper: Mapping Finger Motions onto Virtual Arms to Enable Safe Virtual Reality Interaction in Confined Spaces. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (CHI '23, Article 874). Association for Computing Machinery, New York, NY, USA, 1–14.
- [75] Ying-Chao Tung, Chun-Yen Hsu, Han-Yu Wang, Silvia Chyow, Jhe-Wei Lin, Pei-Jung Wu, Andries Valstar, and Mike Y. Chen. 2015. User-Defined Game Input for Smart Glasses in Public Space. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) (CHI '15). Association for Computing Machinery, New York, NY, USA, 3327–3336. <https://doi.org/10.1145/2702123.2702214>
- [76] Sai Vempala, Rogerio Bonatti, Arthur Buckner, and Ashish Kapoor. 2023. ChatGPT for Robotics: Design Principles and Model Abilities. *arXiv:2306.17582 [cs.AI]* <https://arxiv.org/abs/2306.17582>
- [77] Minh Duc Vu, Han Wang, Zhuang Li, Gholamreza Haffari, Zhenchang Xing, and Chunyang Chen. 2023. Voicify Your UI: Towards Android App Control with Voice Commands. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 7, 1, Article 44 (March 2023), 22 pages. <https://doi.org/10.1145/3581998>
- [78] Uta Wagner, Mathias N. Lystbæk, Pavel Manakhov, Jens Emil Sloth Grønbaek, Ken Pfeuffer, and Hans Gellersen. 2023. A Fitts' Law Study of Gaze-Hand Alignment for Selection in 3D User Interfaces. 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 252, 15 pages. <https://doi.org/10.1145/3544548.3581423>
- [79] Xiangzhi Eric Wang, Zackary P T Sin, Ye Jia, Daniel Archer, Wynonna H Y Fong, Qing Li, and Chen Li. 2025. Can you move these over there? An LLM-based VR Mover for supporting object manipulation. *arXiv [cs.HC]* (Feb. 2025).
- [80] Christian Weichel, Manfred Lau, David Kim, Nicolas Villar, and Hans W. Gellersen. 2014. MixFab: a mixed-reality environment for personal fabrication. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) (CHI '14). Association for Computing Machinery, New York, NY, USA, 3855–3864. <https://doi.org/10.1145/2556288.2557090>
- [81] Johann Wentzel, Alessandra Luz, Martez E Mott, and Daniel Vogel. 2025. MotionBlocks: Modular Geometric Motion Remapping for More Accessible Upper Body Movement in Virtual Reality. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* (CHI '25). Association for Computing Machinery, New York, NY, USA, Article 608, 16 pages. <https://doi.org/10.1145/3706598.3713837>
- [82] Matt Whitlock, Ethan Harnner, Jed R. Brubaker, Shaun Kane, and Danielle Albers Szafir. 2018. Interacting with Distant Objects in Augmented Reality. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 41–48. <https://doi.org/10.1109/VR.2018.8446381>
- [83] Adam S. Williams, Jason Garcia, and Francisco Ortega. 2020. Understanding Multimodal User Gesture and Speech Behavior for Object Manipulation in Augmented Reality Using Elicitation. *IEEE Transactions on Visualization and Computer Graphics* 26, 12 (2020), 3479–3489. <https://doi.org/10.1109/TVCG.2020.3023566>
- [84] Jacob O. Wobbrock. 2019. Situationally-Induced Impairments and Disabilities. In *Web accessibility: A foundation for research* (2 ed.), Yeliz Yjesilada and Simon Harper (Eds.). Springer, London, England, Chapter 5.
- [85] Haijun Xia, Michael Glueck, Michelle Annett, Michael Wang, and Daniel Wigdor. 2022. Iteratively Designing Gesture Vocabularies: A Survey and Analysis of Best Practices in the HCI Literature. *ACM Trans. Comput.-Hum. Interact.* 29, 4 (May 2022), 1–54.
- [86] Momona Yamagami, Sasa Junuzovic, Mar Gonzalez-Franco, Eyal Ofek, Edward Cutrell, John R. Porter, Andrew D. Wilson, and Martez E. Mott. 2022. Two-In-One: A Design Space for Mapping Unimanual Input into Bimanual Interactions in VR for Users with Limited Movement. *ACM Trans. Access. Comput.* 15, 3, Article 23 (jul 2022), 25 pages. <https://doi.org/10.1145/3510463>
- [87] Momona Yamagami, Claire L. Mitchell, Alexandra A. Portnova-Fahreva, Junhan Kong, Jennifer Mankoff, and Jacob O. Wobbrock. 2024. Customized Mid-Air Gestures for Accessibility: A \$ B Recognizer for Multi-Dimensional Biosignal Gestures. *arXiv preprint arXiv:2409.08402* (2024).
- [88] Momona Yamagami, Alexandra A. Portnova-Fahreva, Junhan Kong, Jacob O. Wobbrock, and Jennifer Mankoff. 2023. How Do People with Limited Movement Personalize Upper-Body Gestures? Considerations for the Design of Personalized and Accessible Gesture Interfaces. In *Proceedings of the 25th International ACM SIGACCESS Conference on Computers and Accessibility* (New York, NY, USA) (ASSETS '23). Association for Computing Machinery, New York, NY, USA, Article 1, 15 pages. <https://doi.org/10.1145/3597638.3608430>
- [89] Chao-Han Huck Yang, Yile Gu, Yi-Chieh Liu, Shalini Ghosh, Ivan Bulyko, and Andreas Stolcke. 2023. Generative Speech Recognition Error Correction With Large Language Models and Task-Activating Prompting. In *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. 1–8. <https://doi.org/10.1109/ASRU57964.2023.10389673>
- [90] Nima Zargham, Mohamed Lamine Fetni, Laura Spillner, Thomas Muender, and Rainer Malaka. 2024. "I know what you mean": Context-aware recognition to enhance speech-based games. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA.
- [91] Lei Zhang, Jin Pan, Jacob Gettig, Steve Oney, and Anhong Guo. 2024. VRCopilot: Authoring 3D Layouts with Generative AI Models in VR. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology* (Pittsburgh, PA, USA) (UIST '24). Association for Computing Machinery, New York, NY, USA, Article 96, 13 pages. <https://doi.org/10.1145/3654777.3676451>
- [92] Xiaoyi Zhang, Anne Spencer Ross, Anat Caspi, James Fogarty, and Jacob O. Wobbrock. 2017. Interaction Proxies for Runtime Repair and Enhancement of Mobile Application Accessibility. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) (CHI '17). Association for Computing Machinery, New York, NY, USA, 6024–6037. <https://doi.org/10.1145/3025453.3025846>
- [93] Xiaoyi Zhang, Tracy Tran, Yuqian Sun, Ian Culhane, Shobhit Jain, James Fogarty, and Jennifer Mankoff. 2018. Interactiles: 3D Printed Tactile Interfaces to Enhance Mobile Touchscreen Accessibility. In *Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility* (Galway, Ireland) (ASSETS '18). Association for Computing Machinery, New York, NY, USA, 131–142. <https://doi.org/10.1145/3234695.3236349>
- [94] Yuchong Zhang, Bastian Orthmann, Michael C. Welle, Jonne Van Haastregt, and Danica Kragic. 2025. LLM-Driven Augmented Reality Puppeteer: Controller-Free Voice-Commanded Robot Teleoperation. *arXiv:2502.09142 [cs.HC]* <https://arxiv.org/abs/2502.09142>
- [95] Daniel Zielasko, Sebastian Freitag, Dominik Rausch, Yuen C. Law, Benjamin Weyers, and Torsten W. Kühlen. 2015. BlowClick: A Non-Verbal Vocal Input Metaphor for Clicking. In *Proceedings of the 3rd ACM Symposium on Spatial User Interaction* (Los Angeles, California, USA) (SUI '15). Association for Computing Machinery, New York, NY, USA, 20–23. <https://doi.org/10.1145/2788940.2788953>