EXPLORING MID-AIR GESTURES IN HUMAN-COMPUTER INTERFACES

SABA FALLAH

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

June 2024

Abstract

This thesis investigates three hand gesture detection technologies and their integration into everyday tasks for interacting with technology, including text entry, target selection, and gaming. Through three user studies, these technologies have been incorporated and evaluated within novel applications relying on mid-air hand gestures.

The first user study introduces a one-handed mid-air gesture-based text entry method in Virtual Reality with two layouts of a four-key ambiguous keyboard. Five participants took part in a five-day longitudinal study using a computer camera and an open-source gesture detection framework for gesture detection. Their entry speed (wpm) and error rate (%) were recorded and analyzed.

In the second user study, a Leap Motion Controller (LMC) was integrated with a physical keyboard, introducing a novel computer keyboard, LeapBoard, that combines mid-air hand gestures with physical keys. The evaluation compared LeapBoard's point-and-select ability to a touch-based method (using a touchpad) and a mid-air gesture-based method in a target selection task. A user study with 12 participants measured throughput (bps) and error rates (%) across different selection methods, movement amplitudes, and target widths.

The third user study investigates the effects of two types of mid-air hand gesture-based input methods on children's performance, fun, and preference and compares them to a mouse input method. The evaluation was done using a card-matching game on a laptop with 18 children between five and seven. The trial completion time (s), number of selected cards, and children's perception regarding ease of use, likability, and willingness to play the game using each input method again were recorded and analyzed.

Contents

List of Tables

List of Figures

1 Introduction

1.1 Motivation

While the three user studies in this thesis use different technologies and applications of gesture-based inputs, they all have the same goal of evaluating users' performance and perception of mid-air gesture-based inputs.

In human-computer interaction *(HCI)*, gestures refer to deliberate and purposeful physical movements made by users to interact with digital devices or interfaces. Such interactions can be performed through various means, such as the hands, head, eyes, and specialized physical tools such as a stylus. Mid-air hand gestures, also known as air gestures or free-space gestures, refer to hand gestures performed in the air without requiring physical contact with a device or surface, allowing for a fluid and natural interaction [1]. Sensors or cameras capture and interpret these gestures to control and interact with digital devices. An early example of mid-air gestures is Bolt's "Put That There" demonstration [2]. It was conducted in a room with a large projection screen. The user sat in a chair in front of the screen and made mid-air gestures by moving their hand to position objects at different locations on the screen. See Figure 1. A "+" cursor indicated the corresponding mapping of their hand position on the screen, and voice commands were used to create the objects. An early form of space position and orientation sensing technology was mounted on the user's hand or wrist to detect movements.

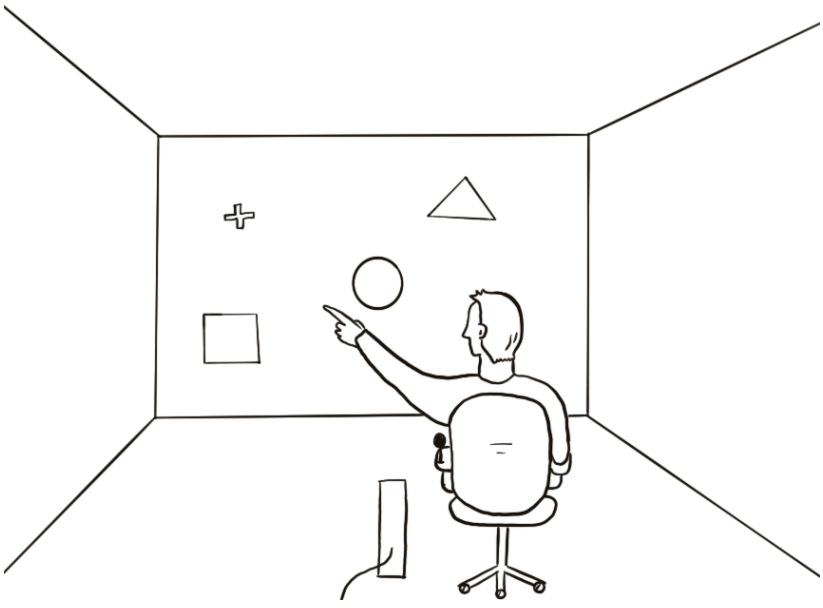


Figure 1: Bolt's "Put That There" demonstration for gesture input with a projector screen.

Gesture-based computer interaction can offer a natural user experience by leveraging gestural

expressions that users are familiar with or can easily learn [3]. By involving users more physically, gesture-based interaction methods can increase the sense of interactivity in a user, especially in activities such as gaming or in augmented or virtual reality [4, 5]. Gesture inputs can also be valuable for individuals with mobility impairments. They provide an alternative means of interacting with computers and devices, potentially increasing accessibility and inclusivity in technology [6, 7].

Considering the significance and contribution of mid-air hand gestures, this thesis was motivated to introduce new interaction methods, investigate potential drawbacks – including issues like gesture misinterpretation, complexity in learning gestures, and the cognitive and physical demands they impose – and provide insightful recommendations for enhancing mid-air hand gesture-based interactions in HCI.

1.2 Mid-Air Hand Gesture Detection Technology

The detection and interpretation of mid-air gestures is required for implementing gesture-based interactions. Various devices and technologies have been developed to capture and interpret these gestures, offering diverse solutions for distinct applications. This section provides an overview of the devices and technology used to detect mid-air gestures.

1. Depth Sensing Technology: Depth sensing technology, such as *Microsoft Kinect* and *Intel RealSense*, utilizes various methods to measure the distance from the device to a particular object or body part in its field of view. These methods include:

    - Structured Light: Projecting a pattern of infrared light onto the scene and measuring distortions in the pattern to determine depth.
    - Time-of-Flight (*ToF*) Cameras: Emitting a light pulse and measuring the time it takes to return, providing depth information for each pixel. ToF sensors are commonly found in smartphones and other consumer devices.
    - Stereo Vision: Utilizing two or more cameras to calculate depth by comparing the images from each camera and identifying disparities. AR and VR headsets may incorporate stereo cameras for hand tracking.

2. Radar Technology: Radar systems emit radio waves and detect reflections to determine the location and movement of objects. Radar technology is increasingly being explored for gesture recognition, particularly in automotive applications [8].

3. Wearable Devices: Wearable devices like smart gloves and wristbands often incorporate multiple sensors, like IMUs, Proximity sensors, EMG sensors, etc., to capture and interpret hand and body movements [9].

4. Inertial Measurement Units (*IMUs*): IMUs combine accelerometers, gyroscopes, and sometimes magnetometers to track the orientation and movement of a device or a user's body part. They are commonly found in wearables, such as smartwatches and VR headsets [10].

5. Machine Learning and Computer Vision: Advanced Machine Learning (*ML*) and computer vision algorithms are employed with regular cameras to identify and interpret gestures. These systems can recognize patterns and movements within image data, allowing for more flexible and adaptable gesture recognition. For instance, the *MediaPipe* library, developed by *Google*, leverages machine learning models for visual understanding and gesture recognition capabilities [11].

6. Infrared (*IR*) Sensors: Infrared sensors, such as proximity sensors and passive IR sensors, can detect the presence of objects or body parts in their vicinity. These sensors are often used in applications where proximity sensing is more critical than fine-grained gesture recognition. For example, they are employed in motion-activated lighting systems [12].

7. Electromyography (*EMG*): EMG sensors detect the electrical signals produced by muscle contractions. These sensors are often attached to the skin and sometimes inserted directly into the muscle tissue, to capture muscle movements, which can be interpreted as gestures or commands [13].

8. Acoustic Sensors: Acoustic sensors, like microphones, can capture the sound generated by certain gestures, such as finger snaps or claps. These sensors are used in applications requiring audio-based gesture recognition [14].

1.3 Our Approach

This thesis utilized three mid-air gesture detection technologies to develop novel applications, each examined in separate user studies. Table 1 summarizes these studies.

The first user study introduces a one-handed mid-air gesture-based text entry method in Virtual Reality with two layouts of a four-key ambiguous keyboard. Five participants participated in a five-day longitudinal study using a computer camera and the MediaPipe library for gesture detection. Their entry speed (wpm) and error rate (%) were recorded and analyzed.

In the second user study, a Leap Motion Controller (LMC) was integrated with a physical keyboard, introducing a novel computer keyboard, LeapBoard, that combines mid-air hand gestures with physical keys. The evaluation compared LeapBoard's point-and-select ability to a touch-based method (using a touchpad) and a mid-air gesture-based method in a target selection task. A user study with 12 participants measured throughput (bps) and error rates (%) across different selection methods, movement amplitudes, and target widths.

Table 1: Summary of the user studies in this thesis.

User Study	Task	Environment	Gesture Detection Technology	Software	Number of Participants
1	Text entry	Virtual Reality	Computer camera, MediaPipe v0.7 library	H4VR virtual ambiguous keyboard	5
2	Point-and-select	Desktop	Leap Motion Controller with Orion SDK v3.2.1	GoFitts	12
3	Children's game	Desktop	Intel RealSense depth camera (D435), MediaPipe v0.7 library	Animal match-up game	18

The third user study investigates the effects of two types of mid-air hand gesture-based input methods on children's performance, fun, and preference and compares them to a traditional mouse input method. The evaluation was done using a card-matching game on a laptop with 18 children between five and seven. The trial completion time (s), number of selected cards, and children's perception regarding ease of use, likability, and willingness to play the game using each input method again were recorded and analyzed. The number of participants used in each study was selected after reviewing related work and was chosen to be close to the numbers used in those studies. In longitudinal studies, often fewer participants are recruited because they are tested in multiple sessions. Hence, the first user study chose a smaller group of participants than the similar studies. Below, detailed information regarding the gesture detection technology employed is provided.

1.3.1 MediaPipe

MediaPipe is an open-source framework developed by Google that has gained significant attention in computer vision and real-time perceptual computing. MediaPipe Gesture Recognizer uses ML models to provide mid-air hand gesture detection and tracking capabilities for Android, Python, and web applications. It detects 21 landmarks on a hand and allows for defining and detecting gestures using those landmarks; see Figure 2. MediaPipe is optimized for real-time processing, making it well-suited for interactive applications.

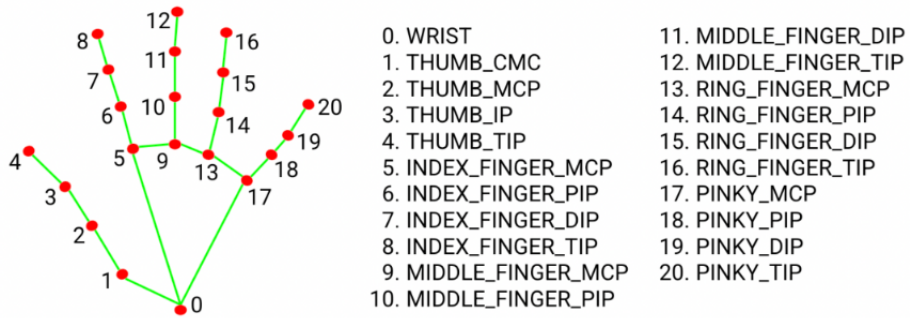


Figure 2: MediaPipe's detected landmarks on each hand.[8]

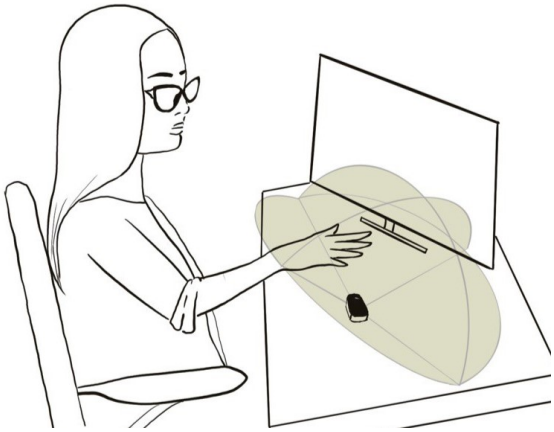


Figure 3: Leap Motion Controller interaction area: 25 to 60 cm above the controller, 140 ° horizontal, and 120 ° vertical.

1.3.2 Leap Motion Controller

The original Leap Motion Controller (LMC)[9] from Ultraleap is a 3D motion-capturing device with two infrared cameras and multiple LEDs. The device's compact size and light weight offer the flexibility to position it in various locations, provided the user's hands are a trackable distance from the device. The controller tracks up to two hands, most optimally from 25 to 60 cm from the device. LMC covers 140 degrees horizontally and 120 degrees vertically. See Figure 3.

LMC employs a form of IR stereo imaging, where infrared LEDs illuminate the hands, and stereo cameras capture the infrared light to create a three-dimensional model of the hands and fingers, aligning it with IR sensor technology.

Leap Motion comes with a software development kit (*SDK*) for hand-tracking that uses the data captured by the controller to model the underlying structure of the hand at the level of the joints

[8] https://developers.google.com/mediapipe/solutions/vision/hand_landmarker
[9] https://leap2.ultraleap.com/leap-motion-controller

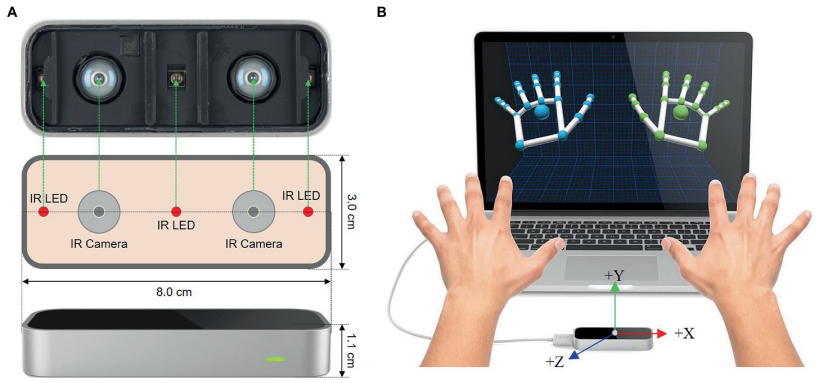


Figure 4: Leap Motion Controller showing (a) schematic view and (b) 3D interaction using the hands [15].

and bones, see Figure 4. It enables gesture recognition through precise tracking of hand and finger movements in a 3D space.

The SDK includes four predefined gestures: circle motion, key tap, screen tap, and swipe. While predefined gestures are readily available, developers can customize them to suit their application's requirements by adjusting parameters like recognition thresholds, directions, or sensitivity. Other data, such as pinch data, hand grab data, finger distances, etc., are available to define gestures using the SDK. Additionally, the data obtained from the Leap Motion SDK can be used to train machine-learning models to define gestures.

1.3.3 Intel RealSense Depth Camera

RGB-D cameras are imaging devices that capture the depth information in addition to a scene's colour (*RGB*). The *Intel RealSense* depth camera D435, part of the Intel RealSense D400 series of cameras, is an RGB-D camera that incorporates an RGB module, a stereo camera consisting of two cameras dedicated to depth perception, and an infrared projector. See Figure 5. It uses a ToF approach to get the depth data from 0.3 m to 3 m. It covers a field of view of $87^\circ \times 58^\circ$. It works at a speed of 90 fps, and the error is less than 2% at 2 m. The camera is connected to the computer using a USB cable.

1.4 Research Contribution

With a overarching objective of introducing new gesture-based interaction techniques, examining possible limitations, and offering informed suggestions to enhance mid-air hand gesture-based interactions in HCI, each user study focused on exploring a specific task in interacting with technology using a different gesture recognition technology. The contributions and objectives of each study are outlined as follows:

Figure 5: The Intel RealSense D435 camera.

1.4.1 First User Study

Since its emergence in the 1970s [16], virtual reality (VR) has been the focus of considerable research. Rapid and continuous improvements in VR have led to extensive commercial use, with applications in gaming, learning, design, marketing, group meetings, etc. With these applications, an inevitable task in VR is text entry.

Text entry is one of the main interaction methods in computing. Various methods have been developed and evaluated on desktop environments, but less research has occurred in a virtual setting. Typically, in VR, text entry uses a Qwerty soft keyboard and hand-held controllers [17, 18]. Using hand-held devices such as controllers might be difficult for some users, especially the ones with motor disabilities [19]. An alternative is hand wearables, such as gloves [20]. However, hand wearables can cause discomfort and are costly. Another possibility is hand gesture recognition using a camera [20]. This is the focus of the present research.

While most text entry methods in VR rely on both hands, one-handed entry is also possible. Some current one-handed text entry methods in VR rely on point-select procedures. The user moves a pointer to the desired key and performs a select operation [21, 22]. Another method is to swipe a finger or a controller across a soft keyboard to make a unistroke gesture [23]. These methods involve more extensive hand motions, which require more space and can cause hand fatigue [24]. One-handed text entry in VR is also done by hand-gesture recognition, where gestures are assigned to characters [25, 26]. Remembering hand gestures and their assigned characters can affect learning time and entry speed.

A Qwerty keyboard has 26 keys just for the English alphabet. Each key must be sized to allow for easy and accurate selection. A keyboard that offers keys with more than one letter per key,

also known as an ambiguous keyboard, presents fewer keys and has the benefit of occluding less space. An ambiguous keyboard can also rely on fewer hand gestures since there are fewer keys to navigate.

Our proposed method is for one-handed gesture-based text entry in VR, H4VR, a four-key soft keyboard based on Huffman codes [27]. Although not studied in VR before, earlier research using this keyboard achieved entry speeds of 20.4 wpm using a game controller [28], 14.0 wpm using finger switches in wearable computing [29], and 15.0 wpm using finger taps on a smartphone [30]. This user study evaluates the entry speed and error rate of two layouts offered by this keyboard through a longitudinal user study.

1.4.2 Second User Study

Research utilizing the LMC SDK for gesture detection has extensively explored mid-air gesture-based applications in target selection tasks, employing methodologies such as multi-directional tapping experiments (ISO 9241 series). A consistent finding is that mid-air gesture-based interaction using an LMC falls short in target selection tasks compared to traditional input devices like mice or touchpads. The hypothesis of our investigation is centred on the notion that a plausible contributing factor to this observed shortcoming lies not in pointing at the targets but in selecting the targets.

Detecting mid-air gestures as selection commands (e.g., clicks) can suffer from issues like lag, gesture misclassification, hand fatigue for performing the gesture, unintended hand movements leading to mispointing, etc. LeapBoard utilizes mid-air hand movement for pointing but does not rely on gesture-based selection methods. Instead, it opts for the convenience and precision of using keyboard keys. This study examines the performance of LeapBoard compared to touch-based and touchless methods in a target selection task.

1.4.3 Third User Study

The third user study investigates children's performance and experience while playing a game on a laptop using three different input methods. The investigated input methods include two one-handed mid-air gesture-based techniques and a mouse. The game consists of matching two pairs of cards by selecting them one at a time.

Children are substantial users of digital devices, with use increasing during the COVID-19 pandemic [31]. They use devices for different purposes, such as entertainment, education, and communication. Playing games is one of children's primary use cases for digital devices. When designed well, games have the potential to offer an engaging and enjoyable experience that supports learning, cognitive growth, skill development, social interactions, physical activities, and healthy

behaviours in young children [32].

Studies have shown that fun has been linked to increased engagement with learning technologies and learning activities [33, 34, 35, 36, 37], and experiencing fun has been associated with positive effects on learning outcomes [38, 39, 34, 40, 35]. Furthermore, neuroscience provides biochemical evidence supporting the beneficial impacts of fun within the learning environment and on the learning process [41].

Various methods have been developed to measure children's fun while interacting with digital devices. One tool in the child-computer interaction (CCI) field is the Fun Toolkit, which consists of three tools – Smileyometer, Fun Sorter, and Again Again Table – developed to measure children's fun and preference for technology [42]. It is designed for children aged four and above.

Children often use gestures as a natural and intuitive way to interact with devices. Gestural interaction, whether through touch-based gestures on screens or mid-air gestures using motion-sensing technology, is increasingly common in applications designed for children.

This study implements and compares two sets of mid-air gestures on children's performance and fun – an aspect that has not been sufficiently explored in previous studies. The evaluation uses a game (see section 2.4) and compares the gestures with a mouse input. As suggested by Read and MacFarlane [43], there is no real point in using a Smileyometer and an Again Again table as both measure the same construct, and the Again Again table seems to have more validity with younger children. This study uses the Fun Sorter and the Again Again table from the Fun Toolkit to measure children's fun. The Fun Sorter inquires about the ease of use and likability of the input methods. Children's performance is studied using quantitative measures.

Space 44	e 22	t 33	a 34	o 31	i 24	n 23	s 42	h 21	r 433
l 411	d 432	c 414	u 413	f 412	m 434	w 324	y 323	p 322	g 321
b 4314	v 4313	k 4312	x 43114	j 43113	q 43112	z 43111			

Figure 6: H4VR character codes generated from base-4 Huffman codes.

2 Software Systems, Implementations and Theories

2.1 H4VR Design

With H4VR, base-4 Huffman encoding generates non-redundant codes for the English alphabet using the character frequency distribution of a corpus. The codes consist of four digits (nominally, '1', '2', '3', and '4'). See Figure 6. Frequent characters (e.g., 'e' and 't') are assigned shorter codes, whereas less frequent characters (e.g., 'q' and 'z') are given longer codes. Note in Figure 6 the codes for the nine most-common characters (top row) have length two. Using Huffman coding, the layout of keys on an ambiguous keyboard can be optimized based on the frequency of use of the character. Huffman encoding helps to reduce the number of keys on the ambiguous keyboard while minimizing the number of selections required to choose a character.

An analytic metric to capture the efficiency of a text entry method is KSPC for keystrokes per character [44]. The equivalent here is GPC for gestures per character. GPC is the average number of gestures to enter a character of text in a given language using a given input method. It is a weighted average computed using a letter-frequency list from a corpus. For English and the 27 symbols in Figure 6, GPC = 2.32 for H4VR. No code is a prefix to another code, meaning delimiting actions between codes is unnecessary. The code set is easily extended to include digits, punctuation, symbols, and commands.

H4VR employs two layouts for evaluation, *flat* and *cross*, each associated with its unique set of hand gestures for key selection. The hand gestures for both keyboard layouts are detected by a computer camera.

In the flat layout, the keys are arranged next to each other; see Figure 7. The cross layout is the original H4-Writer layout with the key positions forming a cross. Three fields are displayed at the top of the keyboards: presented text, transcribed text, and keys. The presented field shows the randomly selected phrase to be entered. The transcribed field displays the phrase that is being entered. The keys field shows the code being entered. The displayed characters on the keys change according to the selected key.

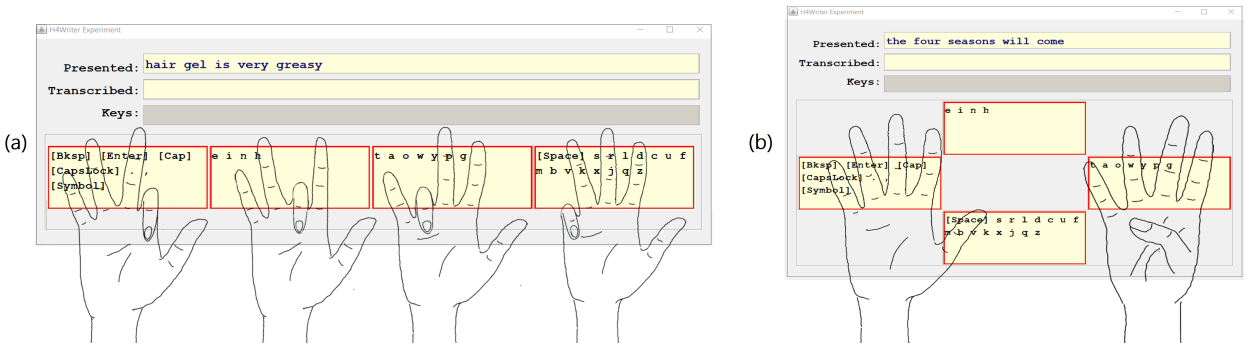


Figure 7: H4VR keyboards and selection methods. (a) Flat layout and code-based selection. (b) Cross layout and virtual mouse selection.

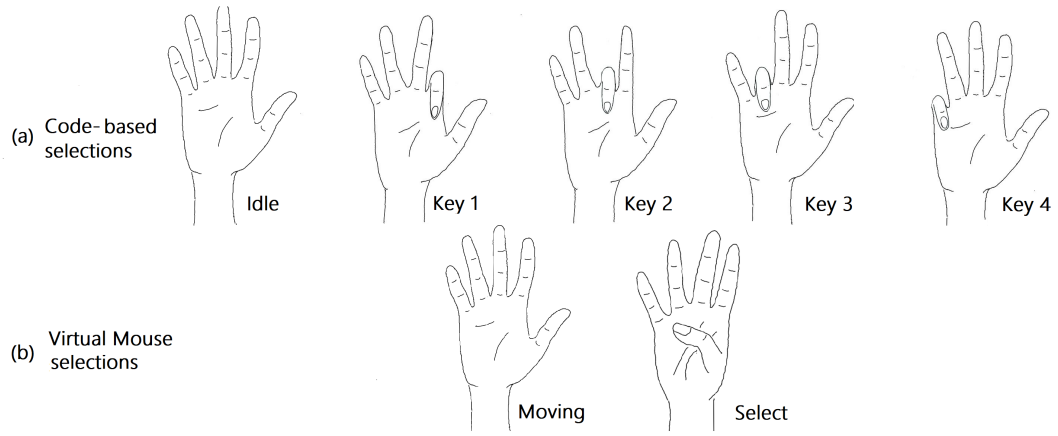


Figure 8: Selection methods for (a) H4VR flat keyboard, and (b) H4VR cross keyboard.

2.1.1 Flat keyboard (with code-based selection)

The flat keyboard uses a selection method called *code-based selection*. Code-based selection allows the user to select characters based on their associated code. The codes assigned to the characters are made up of digits (’1’, ’2’, ’3’, ’4’). Refer to Figure 6. With the palm facing the camera, the user can make hand gestures to select the desired key. The index, middle, ring, and pinkie finger should be closed to enter digits ’1’, ’2’, ’3’, or ’4’, or in other words, select the first, second, third, or fourth key, respectively. An open hand is an idle state where the user is not entering a code.

Figure 8a shows the gestures for the code-based selection. This method does not require positioning a pointer on the key, eliminating the need for hand motions. The flat design of the keyboard makes remembering gestures easier since the gestures were selected to imitate how a person presses a physical key by bringing down the corresponding finger.

The code-based selection can not be used on the H4VR cross keyboard; similarly, the virtual mouse selection (discussed next) can not be used on the H4VR flat keyboard.

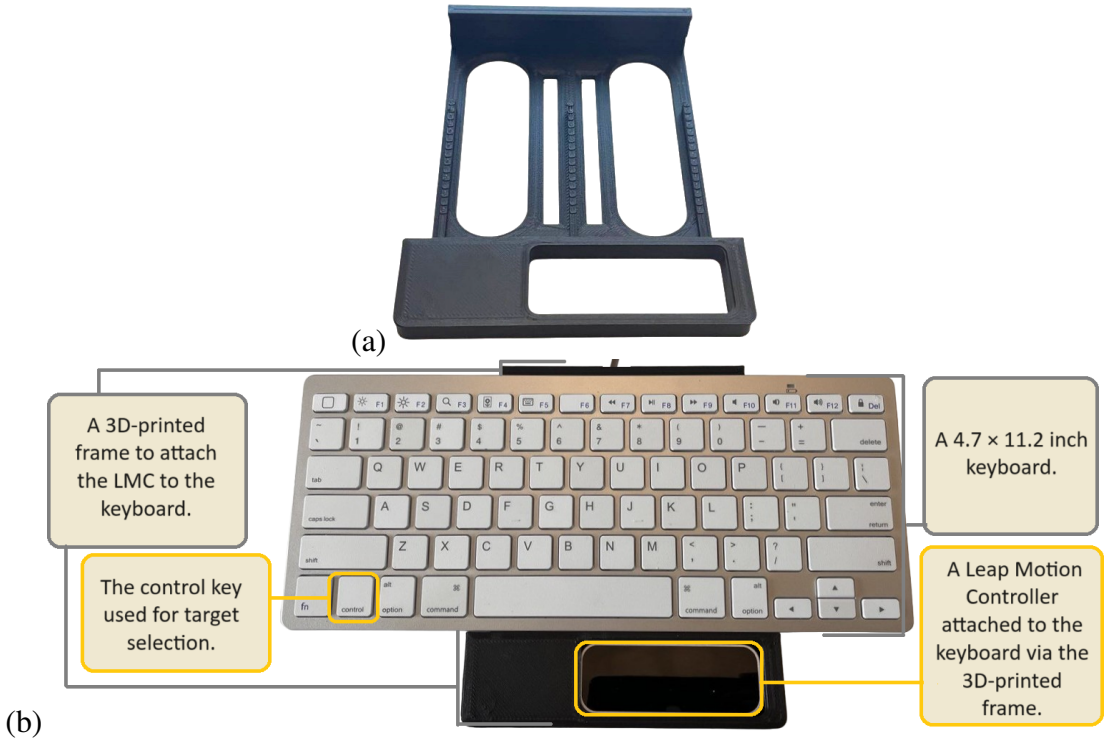


Figure 9: LeapBoard. (a) 3D printed frame. (b) Components in position.

2.1.2 Cross keyboard (with virtual mouse selection)

The cross keyboard uses a different gesture-based key selection method called *virtual mouse selection*. With the palm facing the camera, the movement of the hand is tracked, and a cursor moves accordingly on display. The user's hand is treated like a virtual mouse. With up/down and left/right movements of their dominant hand, the user places the cursor on the desired key. A selection happens when the user closes their thumb. See Figure 8b. The cross layout of the keyboard reduces the hand movement required to position the cursor on the desired key as opposed to the layout used in the flat keyboard as it reduces the distance between the keys.

2.2 LeapBoard Design

The LeapBoard consists of a 4.7 × 11.2 inch keyboard with an LMC attached to it via a 3D-printed frame. The LMC is located at the bottom of the keyboard as an alternative to a traditional touchpad. The LMC and frame height are the same as the keyboard. See Figure 9.

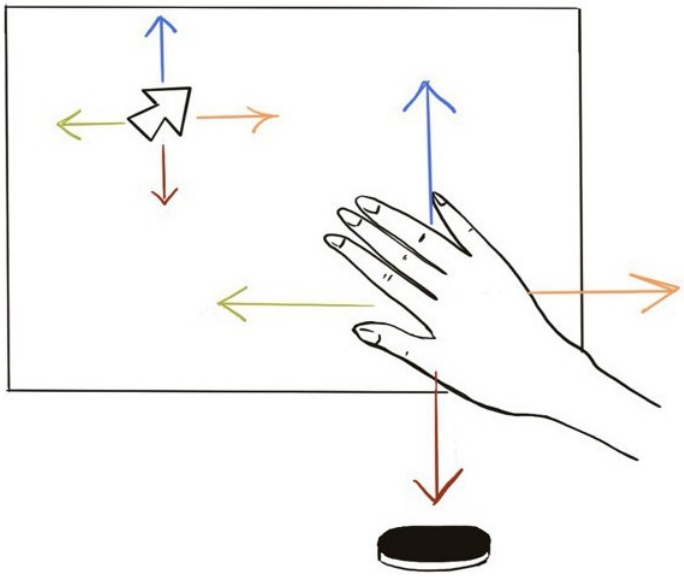


Figure 10: The mappings between hand movements above the Leap Motion Controller and cursor movement on the screen.

2.2.1 Cursor Movement

Hand movement above the LMC is tracked and is used to move the cursor on the screen. The user moves their dominant hand about 6 to 13 inches above the LMC to position the cursor on the target. Up, down, left, or right movement of the hand moves the cursor up, down, left, or right, respectively. See Figure 10. This offers a natural mapping between the hand and cursor movement.

In practice trials, it was observed that some participants moved their hands forward and backward above the LMC, which did not move the cursor. The users quickly corrected the movement. This could suggest that participants experienced a form of overlearning, wherein a new extent of mapping was perceived as intuitive or natural. The LMC Orion SDK v3.2.1 is used in Python to track the palm of the user's hand and to move the cursor on the screen.

2.2.2 Target Selection

While cursor movement depends on touchless dominant hand movement above the LMC on the LeapBoard, target selection uses the non-dominant hand. In our current implementation, selection is done by pressing a physical key on the keyboard attached to the LMC. Other methods are possible depending on the context.

Non-dominant hand selection explores the idea of combining touch- or key-based selection methods with mid-air gesture inputs. The key facilitating a click can be selected in the Python

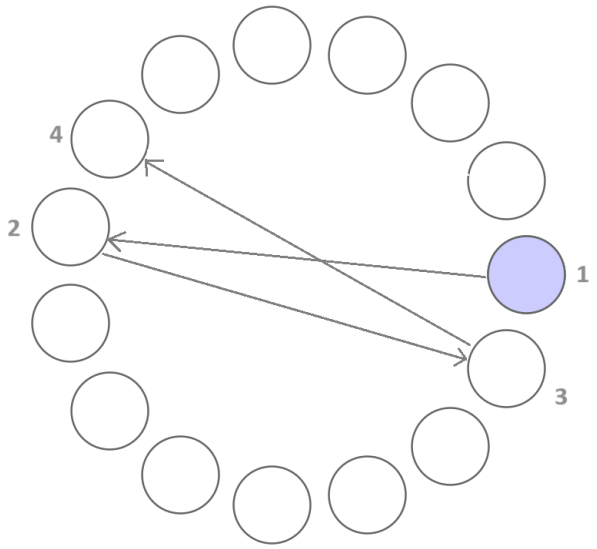


Figure 11: The 2D Fitts' law task in ISO 9241-9.

program to suit different applications. Additional keys on the keyboard can be selected to add more functionality, such as right-clicking, deleting the selected target, etc. This study explores using the control key on the keyboard for clicking, as shown in Figure 9.

It is also possible for selection to occur in some other manner, such as using a surface electromyography (sEMG) interface [45, Figure 3.7] or a piezo-electric sensor positioned, for example, under a headband [45, Figure 3.8]. Such selection possibilities may be important in accessible computing or if the non-dominant hand is busy performing a different task.

2.3 Fitts' Law and ISO Testing

Fitts' law is a well-established method for assessing target selection in computing systems [46, 47, 48]. In the 1990s, the method was incorporated into the ISO 9241-9 standard (revised in 2012 as ISO 9241-411) for evaluating non-keyboard input devices [49]. The primary dependent variable is Fitts' throughput [50].

Fitts' law experiments generally consist of a 2D multi-directional tapping test. Targets of width W are evenly distributed around the circumference of a layout circle. See Figure 11. Participants sequentially choose targets by moving across and around the circle, beginning and ending at the initial target.

Each movement covers an amplitude A, representing the diameter of the layout circle.[10] A trial

[10]Note: Preferrably, an odd number of targets are positioned around the layout circle. This ensures the movement amplitude is the same from each target to the next target. It is also the case that the movement amplitudes are slightly less than the diameter of the layout circle.

is defined as a single target selection task. A sequence is a series of trials to select all the targets around the layout circle. Sequences are configured to cover a range of index of difficulties (*IDs*) in bits/second (bps), with *ID* calculated as follows:

$$ID = \log_2\left(\frac{A}{W} + 1\right) \tag{1}$$

The movement time (*MT*) is measured for each trial and is averaged over each sequence. It is then used to calculate throughput (*TP*). The main performance measure in ISO 9241-9 is throughput (TP) in bits/second (bps), which is calculated over a sequence of trials as the *ID-MT* ratio:

$$TP = \frac{ID_e}{MT} \tag{2}$$

The standard specifies calculating throughput using the effective index of difficulty ID_e. The calculation includes an adjustment for accuracy to reflect the spatial variability in responses:

$$ID_e = \log_2\left(\frac{A_e}{W_e} + 1\right) \tag{3}$$

with

$$W_e = 4.133 \times SD_x \tag{4}$$

The term SD_x is the standard deviation in the selection coordinates computed over a sequence of trials. For the 2D task, selections are projected onto the task axis, yielding a single normalized x-coordinate for each trial. For x = 0, the selection was on a line orthogonal to the task axis that intersects the center of the target. x is negative for selections on the near side of the target center and positive for selections on the far side. The factor 4.133 adjusts the target width for a nominal error rate of 4% under the assumption that the selection coordinates are normally distributed. The effective amplitude (A_e) is the actual distance traveled along the task axis. The use of Ae instead of A is only necessary if there is an overall tendency for selections to overshoot or undershoot the target (see [46] for additional details).

Throughput is a potentially valuable measure of human performance because it embeds both the speed and accuracy of the user's responses. Therefore, comparisons between studies are possible, with the proviso that the studies use the same method in calculating throughputs. Figure 12 is an expanded formula for throughput, illustrating the presence of speed and accuracy in the calculation.

2.4 Animal Match-up Game Design

The game designed for this study – Animal Match-up – presents children with a straightforward yet engaging task. In each game trial, four cards (two pairs) are shown to the player facing down. Card

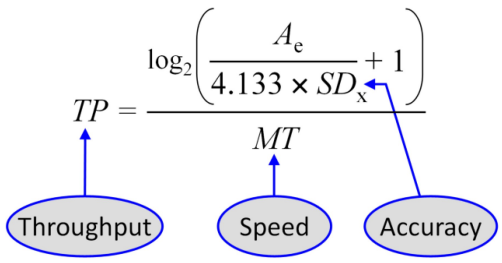


Figure 12: Expanded formula for throughput, featuring speed ($1/MT$) and accuracy (SD_x)

position is randomized. See Figure 13a. Each pair contains the same image of a cartoon animal (the images were obtained from iStock[11]). The player selects cards one at a time. Upon selecting a card, the card is flipped revealing the animal on the opposite side. The player then chooses another card to flip, hoping to match the initial card. If the images match, the pair is solved and remains face-up on the screen. See Figure 13b. If the images do not match, the two selected cards turn face down to hide the animal image. See Figure 13c. The game continues until both pairs are matched; Figure 13d.

While this game requires concentration and memory skills, it does not overwhelm the children, especially when gestures are used to select the cards. The game was developed on the Educaplay[12] platform.

The game is played on a laptop on with three input methods, two using gestures. The gesture methods are one-handed mid-air gestures detected by an Intel RealSense depth camera (D435). When designing the gestures, learnability and memorability were important factors, especially given the age group of the targeted players. It was also important that the gestures were easy to perform. We included two types of gesture-based input methods to study varying numbers of gestures (two and four) and different levels of cognitive power required for each. The three input methods are described below.

2.4.1 Air Mouse

The Air Mouse method uses mid-air hand movement to move the cursor on the screen. With the palm facing the camera, the movement of the hand is tracked and a cursor moves accordingly on the display. With up/down and left/right movements of their dominant hand, the user places the cursor on the card to select. Selection occurs when the player closes their hand (i.e., makes a fist). See Figure 14.

This method relies on moving a cursor with hand movements familiar and intuitive to children.

[11] iStock.com/Turkan Rahimli
[12] https://www.educaplay.com/

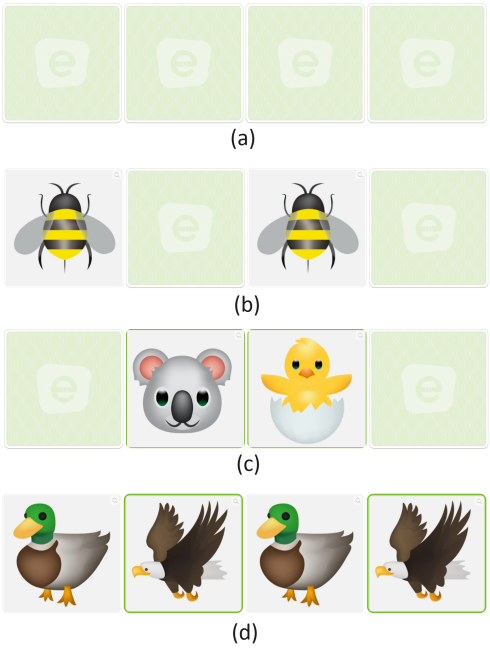


Figure 13: The Animal Match-up game: (a) The game's initial state with two pairs of cards facing down. (b) One pair was matched and remains face up. (c) The selected cards do not match, and the game returns to the initial state. (d) The final state of the game is when both pairs are matched.

The fist gesture for card selection imitates the motion for grabbing items in natural settings, making the interaction intuitive; that is, easily remembered and learned.

2.4.2 Finger Numbers

The Finger Numbers method introduces another set of one-handed mid-air gestures. In this method, there is no need to move the cursor to a position on a card and select it. Instead, the player shows

[13] iStock.com/spukkato

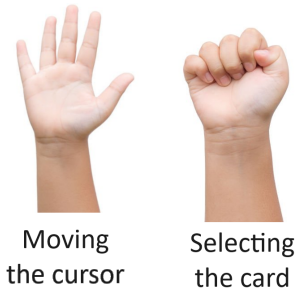


Figure 14: The Air Mouse: An open hand's mid-air movement moves the cursor, and a fist gesture selects the card the cursor is positioned on. (Images from iStock[13])

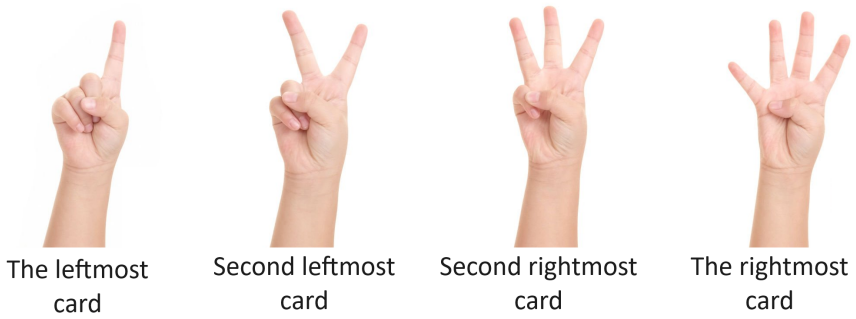


Figure 15: The Finger Numbers: With no need for moving a cursor, showing the number one, two, three, or four with a hand, selects the first, second, third, or fourth card from the left, respectively. (Images from iStock[14])

the number one, two, three, or four with their hands to select the first, second, third, or fourth card from the left, respectively. See Figure 15.

This method aimed to introduce less intuitive gestures that rely more on memory, recall, and cognitive power. The number of gestures also increases from two to four between the Air Mouse and Finger Numbers. The upside is that cursor movement is not required, as with Air Mouse gestures.

2.4.3 Mouse

The mouse input method is included as a baseline condition to compare the performance and fun of using gestures to a traditional and commonly used input method. In trials using the mouse, the player uses a conventional mouse to move the cursor on the screen and position it on the card. A left click on the mouse selects the card and makes the card flip to show the image on it.

[14] iStock.com/3283197d_273

Table 2: Comparison of Text Entry Research in Virtual Reality (VR)

Input Method	Controller-based	Hands	Entry Speed (wpm)	Error Rate (%)	Keyboard	Longitudinal
Controller pointing [17]	✓	2	15.44	0.97	Qwerty	✗
Freehand [17]	✗	2	9.77	7.57	Qwerty	✗
Hand [21]	✗	1	7	2	Qwerty	✗
Controller pointing [21]	✓	1	15	1.8	Qwerty	✗
PizzaText [18]	✓	2	12.26	1.8	Ambiguous - 7 keys	5×10 phrases
HiFinger [51]	✗	1	9.82	6.03	Ambiguous - 6 keys	3×25 min
PinchText [52]	✗	1	12.71	2	Ambiguous - 12 keys	6×10 phrases
RotoSwype [53]	✗	1	14	1	Qwerty	5×20 phrases

3 Literature Review

This section presents an overview of the relevant literature about gesture-based inputs on the discussed topics: text entry, point-and-select, and gaming applications for children.

3.1 Gesture-based Text Entry in Virtual Reality

Table 2 outlines the evaluated techniques based on their reliance on VR controllers, the number of hands engaged, keyboard type, the design of the experiment, and the observed results of text entry speed and error rate.

3.1.1 Text Entry in VR using Qwerty Keyboard

QWERTY keyboards have been adopted as a primary keyboard layout in VR. They leverage the user's familiarity with this layout to deliver an efficient text entry method. The following studies have examined the performance of QWERTY keyboards in VR using both controllers and hand gestures.

Controller-based Text Entry: Speicher et al. evaluated six text entry methods in VR on a Qwerty soft keyboard [17]: head pointing, controller pointing, controller tapping, freehand, discrete, and continuous cursor. They report that the controller pointing method using two handheld controllers to select keys had the best performance with an entry speed of 15.4 wpm. The freehand method using fingers on both hands to type on a virtual keyboard showed an entry speed of 9.77 wpm.

Xu et al. evaluated four one-handed entry methods on a head-mounted display (HMD) using a Qwerty soft keyboard [21]. The methods used a controller, hand, head, and a hybrid of hand and head to select keys. The controller method used ray-casting with a hand-held VR controller, whereby the user cast the ray onto the desired key. A button on the controller was pressed to select the key. This technique resulted in an entry speed of nearly 15 wpm.

Gesture-based Text Entry: In the same study done by Xu et al., the hand method used gesture-based text entry [21]. The user moved their hand to a key, using a closed-palm gesture to select the key. The hand method resulted in an entry speed of about 7 wpm. An error rate of less than 2% was reported for the controller and the hand techniques.

RotoSwype by Gupta et al. [53] allows for swipe typing in VR on a Qwerty soft keyboard with word prediction. A ring equipped with sensors is worn on the index finger to capture the movements and rotation of the user's hand. The ring contains a button used for word selection from the word prediction list. Sixteen participants were examined in a five-day study, where they typed 20 phrases each day. The study shows an entry speed of about 14 wpm with an average error rate of 1% on the last day of the study. Swipe typing leans on hand and wrist mobility which can cause hand fatigue.

3.1.2 Text Entry in VR using Ambiguous Keyboard

Ambiguous keyboards have not been extensively explored in VR. They occlude less space on the display while presenting fewer selection targets which can make them a favourable keyboard in VR. The following studies have examined the performance of ambiguous keyboards with various designs in VR using both controllers and hand gestures.

Controller-based Text Entry: PizzaText by Yu et al. [18] is a round ambiguous keyboard with seven slices appeared on the VR display, with four characters on each. Through thumbsticks on a game controller, the user selects the slice bearing the desired character. Ten participants in a five-day study typed ten phrases each day. The results showed an average speed of 8.59 wpm to 15.9 wpm depending on the users' expertise. The authors reported an error rate of 1.8%.

Gesture-based Text Entry: Jiang et al.'s HiFinger [51] employs a six-key ambiguous keyboard in VR and allows for one-handed wearable text entry. The characters are assigned a two-digit code from the digits 1 to 6. The user moves their thumb toward sensors placed on their fingers, selecting the code corresponding to a character. A three-day study with nine participants resulted in an average text entry speed of 9.82 wpm after 25 minutes of training.

Jiang et al.'s PinchText [52] offers one-handed text entry in VR using pinch gestures and hand positions. It uses an ambiguous keyboard with 12 keys. The position of the user's hand determines the key set (i.e., one of three rows of keys) and different pinch gestures choose specific keys in the selected key set. A tracking system and a sensor detect the hand position. Conductive tape is used on five fingers to detect the pinch gesture. A six-block experiment shows an average entry speed of 12.7 wpm and 11.1 wpm for hand-up vertical (UpV) and hand-down vertical (DownV),

Table 3: Some LMC (Leap Motion Controller) Mid-air Target Selection Gestures.

Gesture Name	Gesture	Number of Hands	References
Screen Tap		One	[60, 61, 62, 63]
Hand Grab		Two	[60, 56, 64]
Tap		One	[62, 65, 66]
Dwell		One	[62]
Pinch		One	[62, 56]

respectively. PinchText requires moving the arm to position the hand to select a key set. This rapid arm movement can cause arm fatigue and needs space to perform the gestures.

Since limited research has been conducted on gesture-based text entry in VR, one of the goals of this study is to address this significant opportunity for further exploration and examination.

3.2 Leap Motion Controller for Gesture-based Point-and-select

Based on a comprehensive literature review, Sluyters et al. [54] note that the majority of LMC publications study either system-defined gestures inherently supported by the LMC or modified gestures that adjust preset static thresholds [55, 56]. The latter they call "opportunistic algorithms." However, more advanced recognition techniques have also been explored. These advanced techniques are based on AI or machine learning algorithms, such as support vector machines (SVM) [57], k-nearest neighbors [58], neural networks algorithms [59], etc.

LMC has been integrated with gesture-based interaction with devices and user interfaces for tasks such as target selection, scrolling, and zooming. A variety of gestures have been mapped to perform various operations. Table 3 summarizes some mid-air gestures assigned to target selection in non-virtual environments using LMC and opportunistic algorithms to define gestures.

Researchers have explored the application of LMC in target selection tasks, often employing Fitts' law and the multi-direction tapping experiment described in ISO 9241-411 [49]. This is detailed in section 2.3. Fitts' law is widely used in such experiments and is instrumental in comparing LMC performance with other pointing and selection devices, including mice and touchpads. Additionally, some studies examined LMC SDK gestures for target selection in other settings.

Dube et al. conducted a Fitts' law experiment with 12 participants comparing four mid-air

target selection methods (push, tap, dwell, pinch) with two types of ultrasonic haptic feedback (select, hover-select) [62]. The system, developed with Unity3D, utilized the Leap Motion Orion 4.0.0 SDK and the Leap Motion Unity Core Assets. This system allowed users to control a cursor on a computer display by moving their hand mid-air, with the Leap Motion Controller tracking hand movements 20 cm above the surface. The LMC SDK was employed to define gestures. A NASA-TLX questionnaire was used to assess the perceived physical and cognitive workload of the selection methods [67]. The results indicated that the mid-air tap gesture was the fastest, most accurate, and one of the least physically and cognitively demanding selection methods. The mid-air pinch gesture was relatively fast but exhibited errors and was physically and cognitively demanding. Intentionally designed to be slow, dwell emerged as the most accurate and the least physically and cognitively demanding method.

Seixas et al. present an experiment with nine participants to compare two mid-air selection gestures (hand grab, screen tap) using the LMC SDK in a Fitts' law task [60]. Throughput, error rate, and additional accuracy measures were analyzed. The mid-air hand-grab gesture involves controlling the pointer position with the dominant hand and selecting by closing and opening the other hand. The mid-air screen tap gesture consists of controlling the pointer position by moving the dominant hand's index finger and selecting by moving the pointing finger toward the screen quickly. The experiment used a within-subjects design with devices/gestures (mouse, screen tap, hand grab), sequences, and blocks as independent variables. Results indicated that the mouse outperformed the LMC regarding movement time, throughput, and error rate. The hand grab gesture exhibited faster movement times than the screen tap gesture, reducing the overall selection time.

In another Fitts' law experiment by Seixas et al. [63], similar findings were reported. The study compared the throughput and error rate of an LMC, mouse, and touchpad and found that the LMC performed poorly compared to the mouse or touchpad. The experiment used a within-subjects design and 12 participants.

Similarly, in a study comparing LMC and a mouse, participants completed a Fitts' law experiment with various target sizes and distances [61]. The experiment involved 48 participants in a within-subjects design. The independent variables were control type (mid-air gestures detected by LMC, mouse), target size (5, 7, 9, & 20 mm), and target distance (50, 100, & 200 mm). Mid-air gestures detected by LMC resulted in lower user productivity, increased finger, wrist, shoulder, and neck fatigue, and lower preference ratings and usability ratings than mouse-based input. The experiments involved participants completing 276 trials with the Leap Motion controller.

The above findings align with the results of research by Bachmann et al. [65], where the performance of mid-air gestures detected by LMC was compared to a mouse in a Fitts' law experiment. The study involved 12 right-handed participants without prior experience with the LMC. Participants performed tasks using both devices, with the analyses focusing on error rate, movement time,

and throughput as dependent variables. The results indicate that the LMC has limitations as an input device for generic computer pointing tasks. The error rate for the LMC was 7.2%, significantly higher than for the mouse (2.3%). Movement times were also longer for the LMC, and throughput was lower compared to the mouse. The LMC did not exhibit a significant practice effect in throughput.

Chatterjee et al. explored integrating eye tracking (using the Eye Tribe tracker) and gesture-based interaction (utilizing an LMC) [56]. The study was conducted on a desktop computer running Windows 8.1. They examined three scenarios to contextualize interactions: a Windows environment with icons and windows, a word processor, and a 3D model viewer for molecules. Each scenario utilized a combination of gaze and gesture-based input for selecting, moving, and manipulating elements. The LMC SDK is used in C++ to define gestures such as mid-air grab gesture (making a fist) and mid-air pinch gesture with the thumb and index finger for target selection, and the bimanual two-finger pinch gesture for zooming. A Fitts' law experiment with 19 participants evaluated the system's performance, comparing different input methods: mouse, touchpad, gaze+dwell, gaze+blink, gesture+dwell, and gaze+gesture. The study assessed the index of performance (*IP*)[15] and the number of time-out trials to evaluate scalability to small targets. Results indicate that gaze+gesture performs comparably to the mouse and touchpad, and outperforms gaze+dwell, gaze+blink, and gesture+dwell.

Zocco et al. propose an augmented environment for efficient command and control (C2) in net-centric military operations using an LMC [66]. The system was integrated into Loki, an advanced C2 system for electronic warfare (EW). The aim was to enhance users' performance and experience by reducing completion time and improving situational awareness. The augmented reality (AR) user interface provides a digitally enhanced view of a real C2 table with the mission area visualized through an optical see-through head-mounted display (HMD) for improved situational awareness. LMC touchless interaction allows users to control a 3D cursor using hand gestures, providing a faster and more natural way to issue commands. Their study involving 12 military experts demonstrated a significant improvement in completion time, number of failures, and failure rate with LMC, particularly in less complex scenarios. However, in more complex situations, the reduction in completion time remained significant, but accuracy was lower compared to touchpad interaction.

Rittitum et al. proposed an approach to enhance team communication during daily scrum meetings using an LMC as an alternative to traditional digital scrum boards controlled with a mouse [64]. Experimental time measurements demonstrated that the proposed hand gestures outperformed mouse controls in tasks like "move task card" and "scrolling one page" and achieved faster task

[15]Note: "Index of performance" in Chatterjee et al.'s report [56, Figure 9] is the same as "throughput" as used in ISO 9241-411 [49] and other Fitts' law studies.

completion times (40.8% and 65.5% faster, respectively).

3.3 Children's Interaction with Digital Devices via Mid-air Hand Gestures

Kauppinen et al. [68] conducted a study featuring the Kinect Stories application, where six children aged four to six engaged with a storybook through gestures. The application utilized NI (natural interaction) Mate software to capture and process real-time motion data from a depth camera. The primary objectives of the study were to determine the suitability of gesture-based input for young users and to assess the error susceptibility of gesture input methods among young children. Users enjoyed the interactive story but were frustrated when the system did not recognize their gestures, which hindered their overall experience with the gestures. Additionally, participants found prolonged standing between interactions uncomfortable.

Renzi et al. [69] used a natural user interface (NUI) and gestures to enhance children's learning of musical notation. They developed a serious game, Touching Notes, employing a hand-detection interface with motion-sensing camera technology. The game features a simple interface with musical note buttons and a scoring system. In two experiments involving a total of 18 children, gesture-based input methods were compared to a traditional mouse interface. The study measured accuracy and observed performance across three trials, revealing that children initially took more time with the gestures, but the time decreased with practice. The results also indicated improvement in the number of correct answers over time. Using a questionnaire, they concluded that all but one of the children preferred using the gesture-based input methods rather than the mouse input, describing their preference because it was more fun and attractive.

Liang et al. [70] designed a gesture-based interactive puppetry system to enhance children's storytelling experiences. The system incorporated a Leap Motion device to capture and interpret hand gestures. A basic set of seven single-hand gestures was developed to engage with a virtual puppetry game. Four children aged 5–8 years were tasked with employing these gestures to manipulate a virtual puppet, guiding its performance in a story and interacting with various elements in a virtual environment to aid narration. Each child underwent five trials. The outcome revealed a notable improvement in children's performance throughout the trials. The time required to complete the game was reduced. The number of wrong hand gestures was also reduced from 3.0 to 0.5 between the initial and final trials.

Alzubi et al. [71] designed an interactive game-based learning system involving 60 children aged 5 to 6 years. Throughout 20 sessions, the children played nine gesture-based learning games using a Microsoft Kinect to capture their movements. The system incorporated three easy-to-remember gestures: select, drag, and drop. The paper suggests that this technology can enhance working memory and foundational mathematical skills in children.

Some researchers used fun measurement tools to evaluate children's feelings toward using mid-air gestures for interacting with devices. Zaina et al. [72] conducted a user study involving 29 children aged 7 to 10 years, aiming to investigate how they experience gesture-based input in a computational thinking (CT) game. The study also compared gesture-based and touch-based inputs for the same game with the touch-based version running on a tablet. The results indicated that children remained more engaged when utilizing hand gestures, leading to increased concentration during the game. The researchers developed five hand gestures that are recognized using an Intel RealSense camera. The self-assessment manikin [73] was used to determine whether children had a positive or negative experience using the methods. The study's findings indicate that children responded positively to both gesture and touch-based interactions, with gestures promoting quicker adaptation and efficient problem-solving.

Rahman et al. [74] studied how the naturalness of gestures in a gesture-based interaction influences children's behaviour and how much children benefit from their use. Participants included 16 children aged between 5 and 6 years. The children played a game with two gesture-based methods: mid-air and touch-based. Microsoft Kinect was used for mid-air gesture detection, and an iPad tablet was used for touch-based gestures. The findings revealed that both mid-air and touch-based gestures had varying degrees of naturalness, with some resembling real-world interactions and others seeming unnatural. Notably, most mid-air gestures were perceived as more natural, mimicking real-world interactions, while fewer touch-based gestures achieved the same level of naturalness. Participants displayed positive body language and expressions, mainly during mid-air gestures. However, challenges arose with touch-based gestures on the iPad, particularly when complex movements were required, leading to disengagement among participants.

During an interview session, a 5-point Likert scale Smileyometer [42] was used to collect children's feedback on their feelings towards the interaction methods. Participants using touch-based gestures generally expressed satisfaction, with 4 participants scoring *very happy* (5) and 11 participants being *happy* (4). In contrast, mid-air gestures received more *very happy* (5) responses (8 participants) but also had 6 participants feeling *OK* (3). The authors believe that if the system can be improved to avoid gesture misinterpretation, more *very happy* results can be obtained for the mid-air gestures. This is because they observed the positive feelings that resulted from using natural gestures were thwarted by many unintended gestures interpreted as command gestures. For instance, when participants adjusted their pants, the body gesture was interpreted as a command. Overall, this study concludes that natural gesture-based interaction benefited the children by enriching their cognitive abilities and making them happy.

Most of the work on this topic investigated the suitability of gestures in children-computer interaction [68] and its effects on learning and thinking [69, 70, 71, 72], with some researchers comparing gestures with touch-based methods such as mouse [69] and tablet [72, 74].

In Rahman et al.'s study [74], the impact of the naturalness of gestures on children's feelings and behaviour was observed and measured by the Smileyometer [42]. However, Read and MacFarlane [43] suggest that the Smileyometer is more useful with older children as younger children's responses tend to skew toward *very happy*. So, the data have little variability.

4 First Experiment - H4VR: One-handed Gesture-based Text Entry in Virtual Reality Using a Four-key Keyboard [16]

4.1 Methodology

The evaluation of H4VR consisted of a longitudinal study with five sessions. The decision to use a longitudinal design was made to allow the participants to familiarize themselves with the gestures and letter locations on the keyboard through sessions spanning several days and to investigate this process. During each session, participants entered 20 phrases selected randomly from the MacKenzie and Soukoreff [76] phrase set. Although the four-key keyboard used for this study allows for entering symbols, uppercase letters, and numbers, the sessions consisted of entering only lowercase letters and a space between words. Due to the limited number of sessions, the primary goal was to familiarize the participants with the position of each character on the keyboard and the codes assigned to them. The number of sessions and phrases for each session was selected based on the similar work reviewed in this paper.

4.1.1 Participants

Three male and two female participants were recruited from the local university (M = 25 years, SD = 0.89). This is fewer participants than in many studies, but such is common in longitudinal studies. Despite the smaller sample size, the total data entries collected for each keyboard (five participants × 20 phrases × 5 sessions = 500) are sufficient to draw conclusions. Two participants had prior experience in text entry on a Qwerty keyboard in VR. None, however, had used ambiguous keyboards in VR and none had worked with keyboards with similar character arrangements as H4VR in desktop or virtual environments. After completing each session, the participants received $20.

4.1.2 Apparatus

The H4VR software implements two keyboard layouts, *flat* and *cross*. See section 2.1 for details. The software was run on Microsoft *Windows 10* on a desktop environment.

Audio feedback as "click" is given when the participant selects a key. Before starting the first session, some participants asked for a view of their hand in the virtual environment. To address this, the image captured by the camera was displayed next to the keyboards during all sessions for all the participants. A Meta *Quest 2* head-mounted display (HMD) mirrored the desktop display, showing the H4VR interface. A 1080p *NexiGo* HD webcam with a resolution of 1920 × 1080

[16] This work was published as a late-breaking work in the *ACM CHI EA '23* [75].

with 30 fps was used to detect hand gestures. *MediaPipe v0.7*, a Python library for hand and finger tracking, was employed to identify hand gestures.

4.1.3 Procedure

Participants engaged in a five-day study over two weeks. The table of character codes (Figure 6) was provided to participants one day before the first session to familiarize them with the codes. Participants were tested individually in a lab. Upon arrival at the lab, participants were instructed to sit comfortably in front of a camera connected to a PC. See Figure 16. The keyboard layouts and key selection methods were explained at the beginning of the first session. They put on and adjusted the HMD to have a clear display and a comfortable VR experience. The HMD mirrored the PC's display and showed the H4VR layout. The participant raises their hand with their palm facing the camera to start typing. The trials began when the participants selected the first character. They were allowed one trial before starting the experiment in each session.

During each session, participants entered ten phrases using the flat keyboard with the code-based selection method and ten phrases using the cross keyboard using the virtual mouse selection method. During each session, the participants could take breaks for up to ten minutes between the trials.

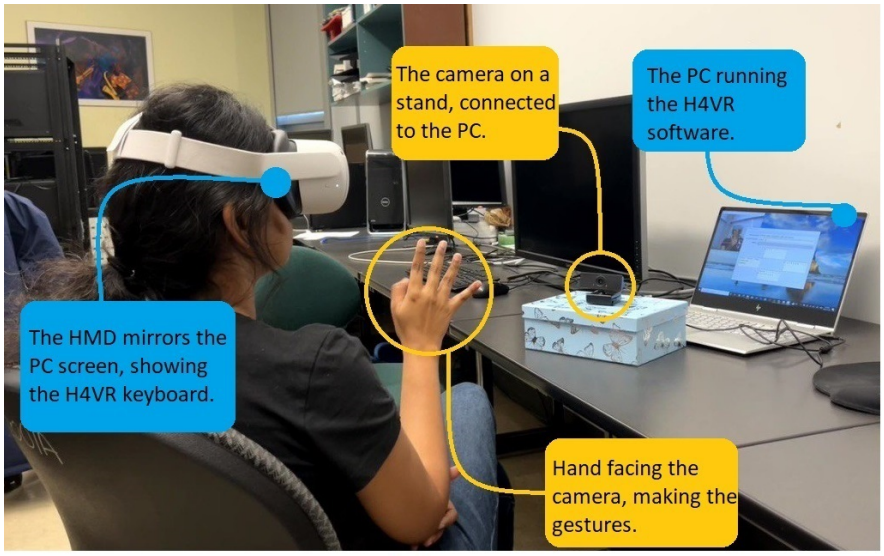
In the first session, all the participants first entered ten phrases using the code-based method and moved on to entering ten phrases using the virtual mouse method. In the second session, to offset order effects, they used the keyboard in reverse order. This swap happened for the rest of the sessions too. The first session took about an hour to complete. The time was less during the following sessions as the entry speed increased. The participant's entry speed and the error rate were recorded for each phrase. Upon completing the last session, they were asked to complete a questionnaire regarding their experience with the keyboards and key selection methods.

4.1.4 Design

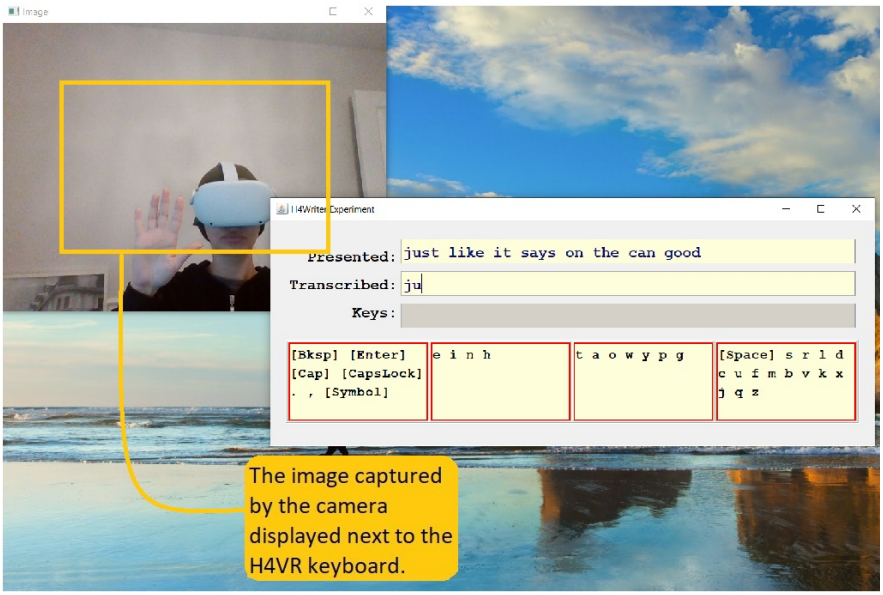
The user study employed a 2×5 within-subjects design using the following independent variables and levels:

- Keyboard (flat + code-based selection, cross + virtual mouse selection)
- Session (1, 2, 3, 4, 5)

The dependent variables were text entry speed (wpm) and error rate (%). In addition, data related to hand fatigue and preference were collected via a custom-made questionnaire administered in the last session; refer to the Appendix. The total phrases entered by each participant was 2 text entry methods $\times$ 10 phrases per method $\times$ 5 sessions = 100.



(a)



(b)

Figure 16: (a) The lab setting. (b) Desktop display of a participant using the flat keyboard.

4.2 Results and Discussion

4.2.1 Entry Speed

The average text entry speed on the cross and flat keyboards was 2.87 wpm and 3.59 wpm, respectively. See Figure 17a. Using an ANOVA, the effect of the H4VR keyboard on entry speed was statistically significant ($F_{1,4} = 23.58$, $p < .01$). The higher average entry speed on the flat keyboard was expected since it eliminates the time required for moving and placing a pointer on a key.

Text entry speed on the cross keyboard started with a mean of 2.19 wpm ($SD = 0.3$) in session 1 and finished at 3.63 wpm ($SD = 0.35$) in session 5 (65.7% increase). Text entry speed on the flat keyboard started with a mean of 2.36 wpm ($SD = 0.29$) in session 1 and finished at 4.57 wpm ($SD = 0.37$) in session 5 (93.6% increase). Figure 17b shows the text entry speed increase over the five sessions for both keyboards. The average text entry speed on the flat keyboard was higher than the cross keyboard for all the sessions. The effect of the session on entry speed was statistically significant ($F_{4,16} = 27.71$, $p < .0001$), thus confirming the expected learning effect with practice. The average improvement in entry speed over the five sessions was also 28% higher in the flat keyboard than the cross keyboard.

Since the keyboards rely on learning the location of each character, it is assumed that more practice can improve the text entry speed as remembering the location of the characters becomes easier. Figure 18 shows the power law of learning for the flat keyboard with entry speed = $2.43n^{0.38}$, where n is the session number. The model suggests an entry speed of about 6 wpm at session 10. The error rate was not made into a model as it is more erratic and less stable than measures of task completion time.

As expected, the session $\times$ keyboard interaction effect was not statistically significant on the entry speed ($F_{4,16} = 2.23$, $p > .05$).[17] This indicates that the learning pattern was about the same for both keyboards.

The H4VR keyboards have fewer keys than the other reviewed studies with ambiguous keyboards. Considering the one-hand entry method used in H4VR, HiFinger [51] and PinchText [52] are the closest to H4VR. HiFinger, with six keys, reported an entry speed of 9.82 wpm after 75 minutes of use. This entry speed is about 73% faster than the H4VR flat keyboard and 92% faster than the H4VR cross keyboard. PinchText with 12 keys reported an entry speed of 12.71 wpm after six sessions of entering ten phrases. The prediction trajectory for the flat keyboard suggests that the expected entry speed on the sixth session is 4.8 wpm, showing a 90% lower entry speed than the PinchText. Although offering a lower entry speed, the flat keyboard does not rely on any attachments to the hand, such as sensors and conductive tape, unlike HiFinger and PinchText.

[17] This is a non-significant result with $(\eta^2) = 0.36$.

[18] Using a linear model, R^2 is lower than the power law model at .9177. Also, it is known that the progression follows the power-law of learning.

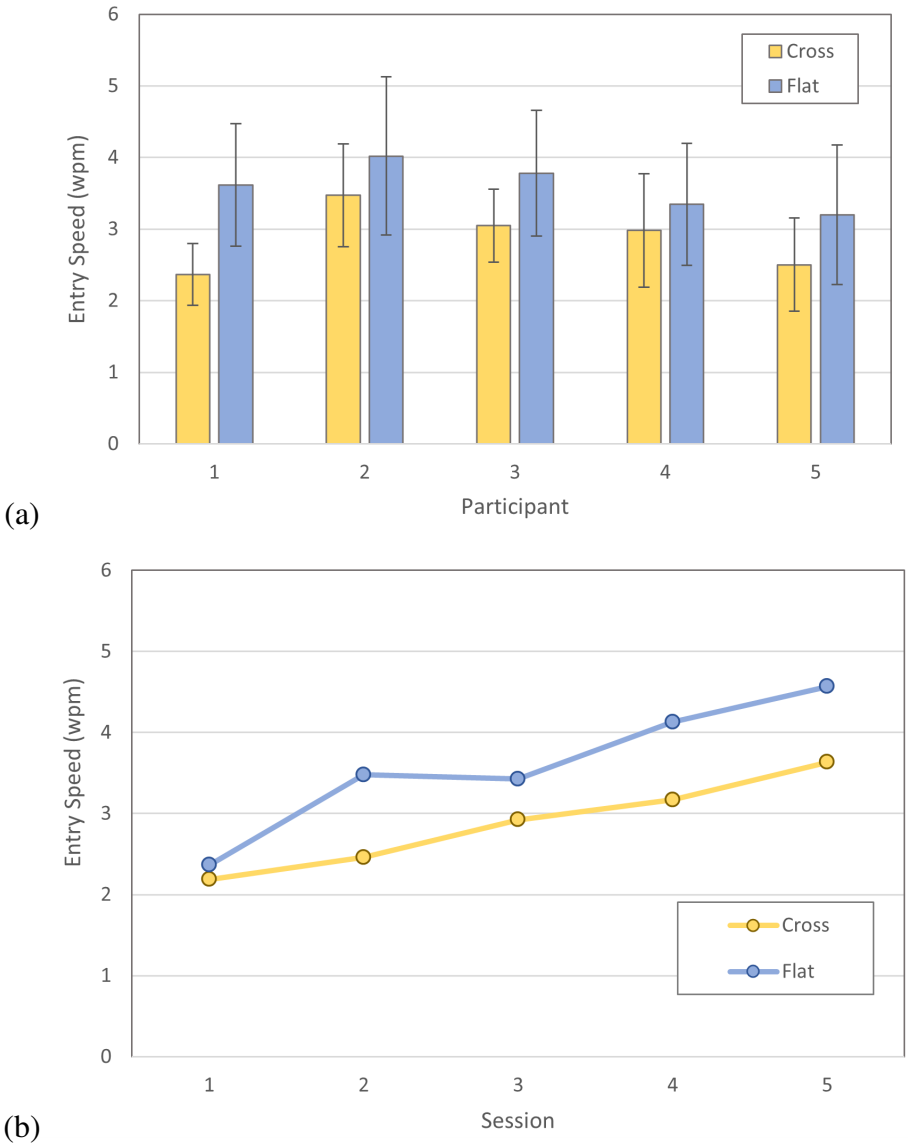


Figure 17: Entry speed (wpm) results: (a) Participants average entry speed (wpm) using H4VR keyboards in five sessions. Error bars show ± 1 *SD*. (b) Entry speed (wpm) by H4VR keyboard and session.

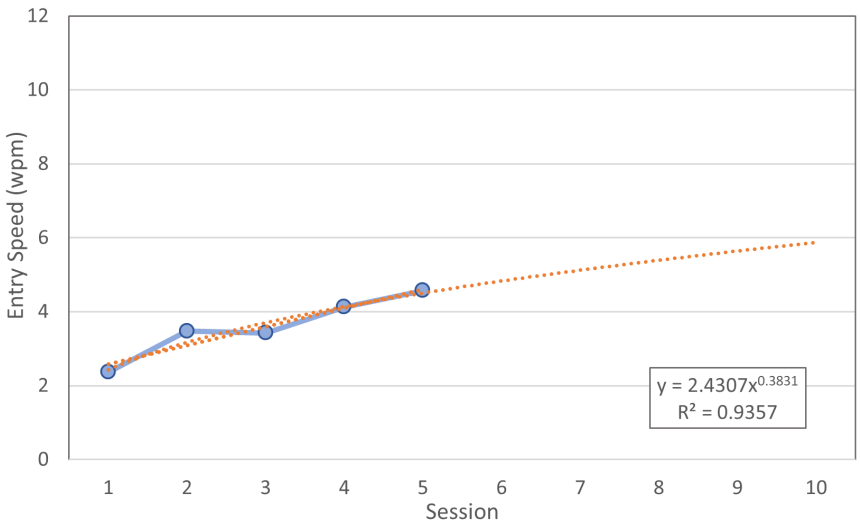


Figure 18: Power model of learning for text entry speed (wpm) on the flat layout over five sessions with a projection to session ten.[18]

4.2.2 Error Rate

The average error rate was generally low for both H4VR keyboards. The average error rate for the cross keyboard was 0.79% (SD = 0.75), while the average error rate for the flat keyboard was 1.82% (SD = 1.42). The difference was statistically significant ($F_{1,4} = 7.72$, $p < .05$). Figure 19a shows the error rate of the keyboards for four sessions. The second session shows a very high error rate for the flat keyboard. This occurred because of a very high error rate with a small number of phrases. High error rates occurred when the user inadvertently selected ENTER prematurely, which resulted in a very high error rate. In Figure 19b, phrases were removed if the error rate was greater than 30%. This resulted in the removal of 9 out of 500 phrases (1.8%). Although the error rate on the last session was the lowest for both H4VR keyboards among other sessions, the effect of the session on error rate was not statistically significant ($F_{4,16} = 1.44$, $p > .05$).

The session $\times$ keyboard interaction effect also was not statistically significant on the error rate ($F_{4,16} = 0.90$, ns). HiFinger utilizes two more keys than H4VR [51]. The error rate of HiFinger at 6.03% was more than twice the H4VR flat keyboard. The error rate of PinchText [52] with 12 keys was similar to the H4VR keyboards at 2%.

H4VR utilizes an ambiguous keyboard with fewer keys than any similar work done in VR. Fewer keys make gesture-based selection easier since the user has to remember only four hand gestures. The gestures are easy to remember as there are only four selection gestures in the flat keyboard and only one selection gesture in the cross keyboard. The small number of gestures relies less on users to recall the gestures and can prevent overwhelming the user. The gestures are also easy to make since they rely on minimum hand movement. A smaller part of the screen is occluded,

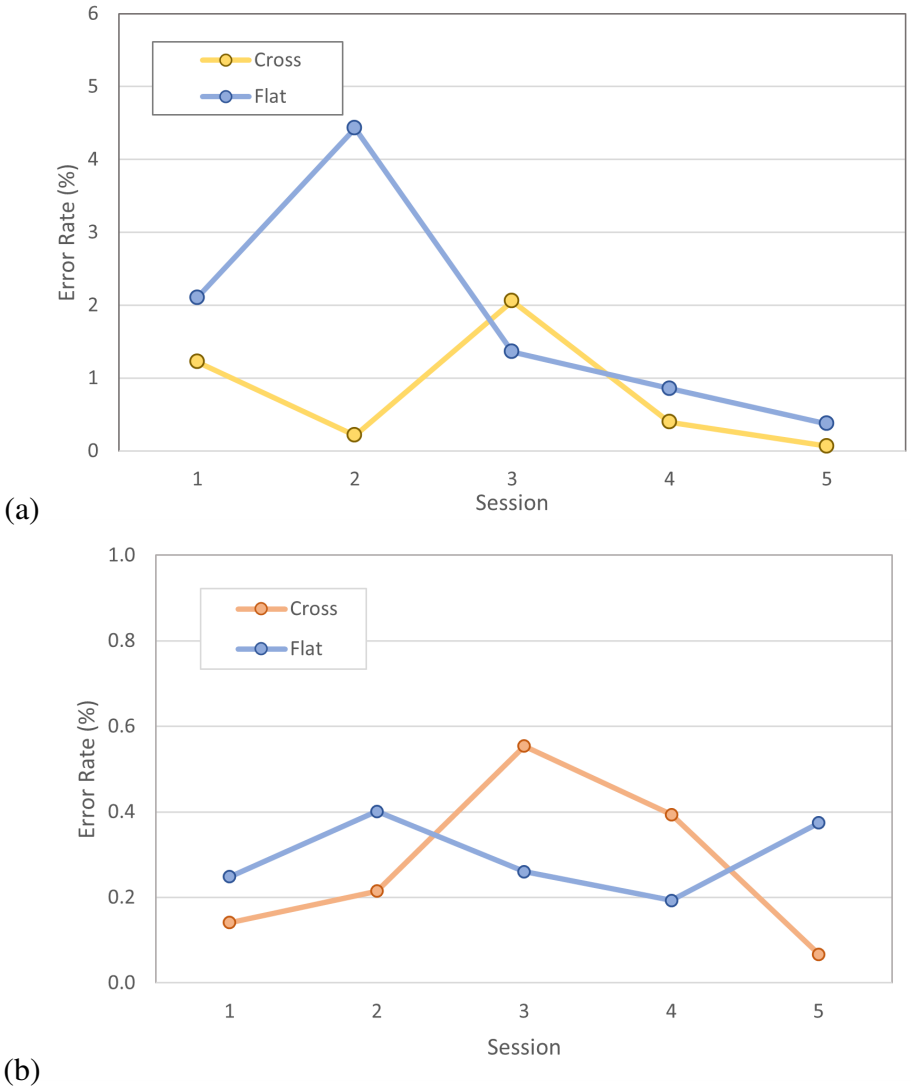


Figure 19: Error rate (%) on cross and flat layouts: (a) including data from all phrases. (b) excluding data from phrases with error rate greater than 30%.

which is one of the main advantages of using an ambiguous keyboard. It also relies on only one hand for text entry, which makes it more valuable to people who might have control of only one hand. The factor limiting higher entry speed during the experiment was likely the time required for the system to detect gestures accurately, particularly when hands were not fully facing the camera. Some participants' feedback highlighted issues with fast double selection on the same key being recognized as one selection, suggesting that hardware enhancements for immediate gesture recognition could further enhance entry speed in the future.

While the condition of this study was to use the H4VR keyboards while receiving visual feedback on the keyboards and the text being entered, the flat keyboard can be utilized for eye-free text entry. By practice, the user can learn the codes assigned to the characters and the need for constant eye contact with the keyboard can be eliminated or reduced. However, this can increase the error rate, as seen in the HiFinger [51], which is an eye-free text entry method.

4.2.3 Hand Fatigue and Preference

In the last session, participants were asked to rank their overall level of hand fatigue for both keyboards. Responses were on a 7-point scale from 1 (*no fatigue*) to 7 (*very fatiguing*). The means were 3.2 and 4.4 for the flat and cross keyboards, respectively, indicating less hand fatigue for the flat keyboard. However, the difference was not statistically significant in a Wilcoxon Signed Ranks test (z = -1.069, $p > .05$). All participants reported that hand fatigue reduced as the sessions progressed. This could be a result of reduced trial duration as the participants got more familiar with the keyboards and the location of the characters. In the cross keyboard, getting familiar with the location of the characters can also reduce unnecessary hand movements during visual search, hence reducing hand fatigue. Four participants reported that they preferred using the flat keyboard, while one preferred cross.

5 Second Experiment - LeapBoard: Integrating Leap Motion Controller with a Physical Keyboard for Gesture-Enhanced Interactions [19]

5.1 Methodology

A user study was conducted to evaluate LeapBoard's performance in a target selection experiment using a Fitts' law task. LeapBoard's performance was compared to a touchless gesture-based selection method using the LMC SDK gestures as well as a touchpad. The goal was to examine how combining mid-air gestures detected by an LMC with physical keys performs compared to touch-based and touchless gesture-based methods.

5.1.1 Participants

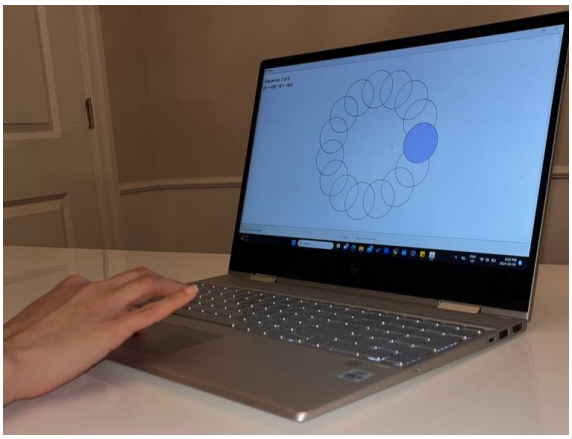
Twelve participants (seven male, five female) were recruited from York university (M = 21.7 years, SD = 2.7). Participants were compensated with $20 for their time. None of the participants had prior experience using an LMC. While a few were familiar with Fitts' law, none had participated in a Fitts' law experiment.

5.1.2 Apparatus

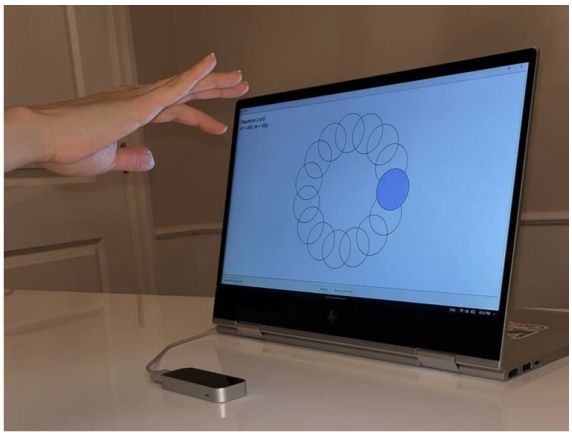
The experiment ran on an HP Envy x360 laptop with a screen resolution of 1536 $\times$ 864 with Windows 11. The GoFitts software was used for the target selection experiment.[20] The following three conditions were implemented for comparison in the user study.

LMC – mid-air gestures and gesture-based selection using a Leap Motion Controller The participant's hand movement above the LMC is tracked and used to move the cursor on the screen. See Figure 20b. The mid-air tap gesture is used for target selection. See Table 3. When the distance of the tip of the middle finger to the tip of the index finger is equal to or more than 40 mm, target selection happens. As noted by Dube et al. [62], the mid-air tap gesture was the most accurate among other explored LMC SDK gestures for target selection. A pilot study before the experiment also confirmed that the mid-air tap gesture was the most accurately detected gesture among other LMC SDK gestures. The LMC Orion SDK v3.2.1 in Python is used in this study to detect the tap gestures and track hands. This method is referred to as the LMC sessions in the following sections.

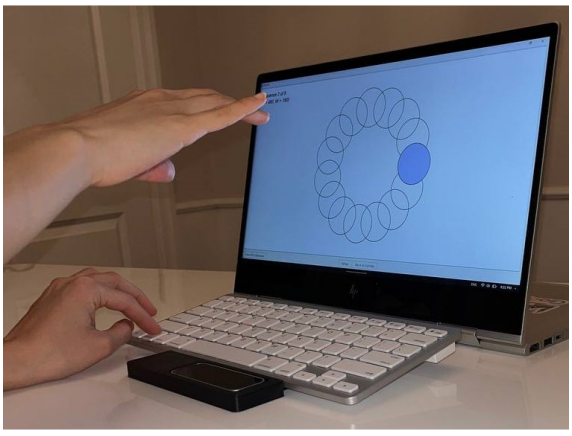
[19] This work is submitted to be published at a conference and is pending results.
[20] https://www.yorku.ca/mack/FittsLawSoftware/



(a)



(b)



(c)

Figure 20: Target selection methods: (a) Touch-based using a touchpad. (b) Mid-air gesture-based using a Leap Motion Controller. (c) Combination of key-based selection and mid-air gestures using the LeapBoard.

LeapBoard – combination of key-based selection and mid-air gestures using the LeapBoard As detailed in section 2.2, LeapBoard uses mid-air hand gestures to move the cursor and the control key on the keyboard for target selection. See Figure 20c.

Touchpad – touch-based selection using a touchpad The laptop's original keyboard with its integrated touchpad was used for the sessions in which the user was expected to use the touchpad. See Figure 20a. Moving one finger on the touchpad moves the cursor on the screen, and a press on the touchpad results in target selection.

5.1.3 Procedure

Participants took part in a roughly one-hour study conducted individually within a lab setting. Upon arrival at the lab, participants were instructed to sit comfortably in front of the laptop. Counterbalancing to offset order effects was performed by dividing participants into three groups, four participants per group. Each group was assigned a different sequence of selection methods according to a 3×3 Latin square.

For the LMC and LeapBoard sessions, the laptop's keyboard was folded back and replaced by the LMC or the LeapBoard to ensure the same screen distance for all sessions. Each participant within these groups completed four sessions for each selection method, with each session comprising nine sequences.

Each sequence presented 15 targets. The movement amplitude and target width conditions were selected at random from the sets A = 120, 240, and 480 pixels and W = 50, 80, and 160 pixels. For each sequence, timing began upon selecting the first target. As sequences proceeded, participants' throughput (TP in bps) and error rate (%) were captured and recorded on a per-sequence basis. Trace data were also recorded for the 2D gesture paths in controlling the cursor.

Before the first session for each selection method, participants received an explanation of the task and the selection methods and performed one session as a practice trial. During the experiment, the participants were allowed to take breaks as they wished. Upon completing all sessions for all selection methods, participants were asked to complete a questionnaire to gather insight on their experiences with the selection methods.

5.1.4 Design

The experiment used a $3 \times 4 \times 3 \times 3$ within-subjects design with the following independent variables and levels:

- Selection method (LMC, LeapBoard, touchpad)
- Session (1, 2, 3, 4)

- Amplitude (120, 240, 480 pixels)
- Width (50, 80, 160 pixels)

Amplitude and width were included as independent variables to ensure the tasks covered a reasonable range of difficulty. With the values used, the index of difficulty ranged from $ID = \log_2(\frac{120}{160} + 1)$ = 0.81 bits to $ID = \log_2(\frac{480}{50} + 1)$ = 3.41 bits.

The dependent variables were throughput (bps) and error rate (%).

The total number of trials was 3 Selection Methods × 4 Sessions × 3 Amplitudes × 3 Widths × 15 targets per sequence = 1,620 trials for each participant, or 12 × 1,620 = 19,440 total trials in the experiment.

5.2 Results and Discussion

5.2.1 Throughput

The grand mean for throughput was 2.60 bps. The breakdown by selection method is presented in Figure 21. By selection method, the highest throughput was 3.55 bps for LeapBoard, followed by 2.26 bps for touchpad, and 1.97 bps for LMC. The difference was statistically significant ($F_{2,22}$ = 40.03, $p < .0001$). It was found that trials with higher widths and lower amplitudes resulted in higher than usual throughput, which appear as outlier points in Figure 21. A Scheffé post hoc test showed a significant difference between LeapBoard and the other two selection methods. This was expected since throughput for LeapBoard was 80.2% higher than LMC and 57.1% higher than with the touchpad.

The breakdown by amplitude and width is presented in Figure 22. Based on individual participants and single sequences, the highest *TP* (throughput) was 7.58 bps for 120 pixels, 6.94 bps for 240 pixels, and 6.37 bps for 480 pixels. An ANOVA test showed that the effect of amplitude on *TP* was statistically significant ($F_{2,22}$ = 61.06, $p < .0001$).

By width, the highest *TP* for single sequences was 7.58 (bps) for 160 pixels, 6.5 (bps) for 80 pixels, and 5.73 (bps) for 50 pixels. An ANOVA test showed the effect of width on *TP* was not statistically significant ($F_{2,22}$ = 1.15, $p > .05$).

An ANOVA revealed that the Selection Method × Amplitude interaction effect was statistically significant ($F_{8,88}$ = 38.12, $p < .0001$). A Scheffé post hoc test revealed that the *TP* of LeapBoard was significantly higher for all amplitudes than the LMC. LeapBoards *TP* was also significantly higher than the touchpad for amplitudes 120 and 240, based on a Scheffé post hoc test.

[21] The outlier points are defined as those that lie beyond the whisker ends, which lie at 1.5 × the interquartile range away from the top or bottom of the box.

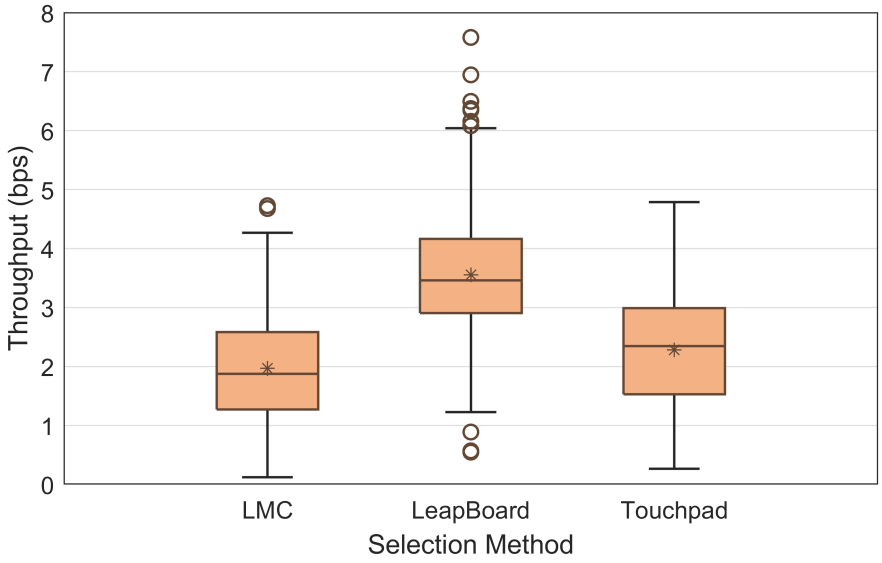


Figure 21: Throughput (bps) by selection method. Asterisks show the mean of the responses.[21]

During the sessions where LMC was used, it was observed that with higher amplitudes, particularly when choosing targets near the bottom of the screen, participants had to position their hands closer to the LMC to accurately place the cursor on those targets. This proximity, approximately 4 inches from the LMC, presented a challenge for detecting the tap gesture. As a result, participants often needed to repeat the tap gesture multiple times for the LMC to detect it. While the tap gesture posed recognition challenges close to the LMC, the hand movement for cursor positioning did not encounter a similar obstacle, resulting in a higher *TP* in the same amplitudes using the LeapBoard.

The significantly higher *TP* of the LeapBoard than the touchpad can be attributed to the mid-air cursor movement and the two-handed approach: selection occurs with a non-dominant hand key-press immediately when the cursor enters the target.

An ANOVA test showed the effect of Selection Method $\times$ Width was statistically significant ($F_{8,88} = 31.84$, $p < .0001$). A Scheffé post hoc test revealed that the *TP* of LeapBoard was significantly higher for all widths than the LMC. The Scheffé post hoc test also revealed LeapBoards *TP* for the width of 160 pixels was significantly higher than the touchpad for all widths. Additionally, LeapBoards *TP* was significantly higher than the touchpad for width 80. Widths 50 and 80 using the LeapBoard resulted in significantly higher *TP*s compared to the width of 160 pixels using the touchpad.

The Selection Method $\times$ Amplitude $\times$ Width interaction was found to be statistically significant ($F_{26,286} = 25.03$, $p < .0001$). Specifically, LeapBoard's *TP* was significantly higher than that of LMC for the combination of amplitude 120 and width 50 pixels. Furthermore, the LeapBoard *TP* was significantly higher over the touchpad for a width of 160 pixels in amplitudes of 120 and

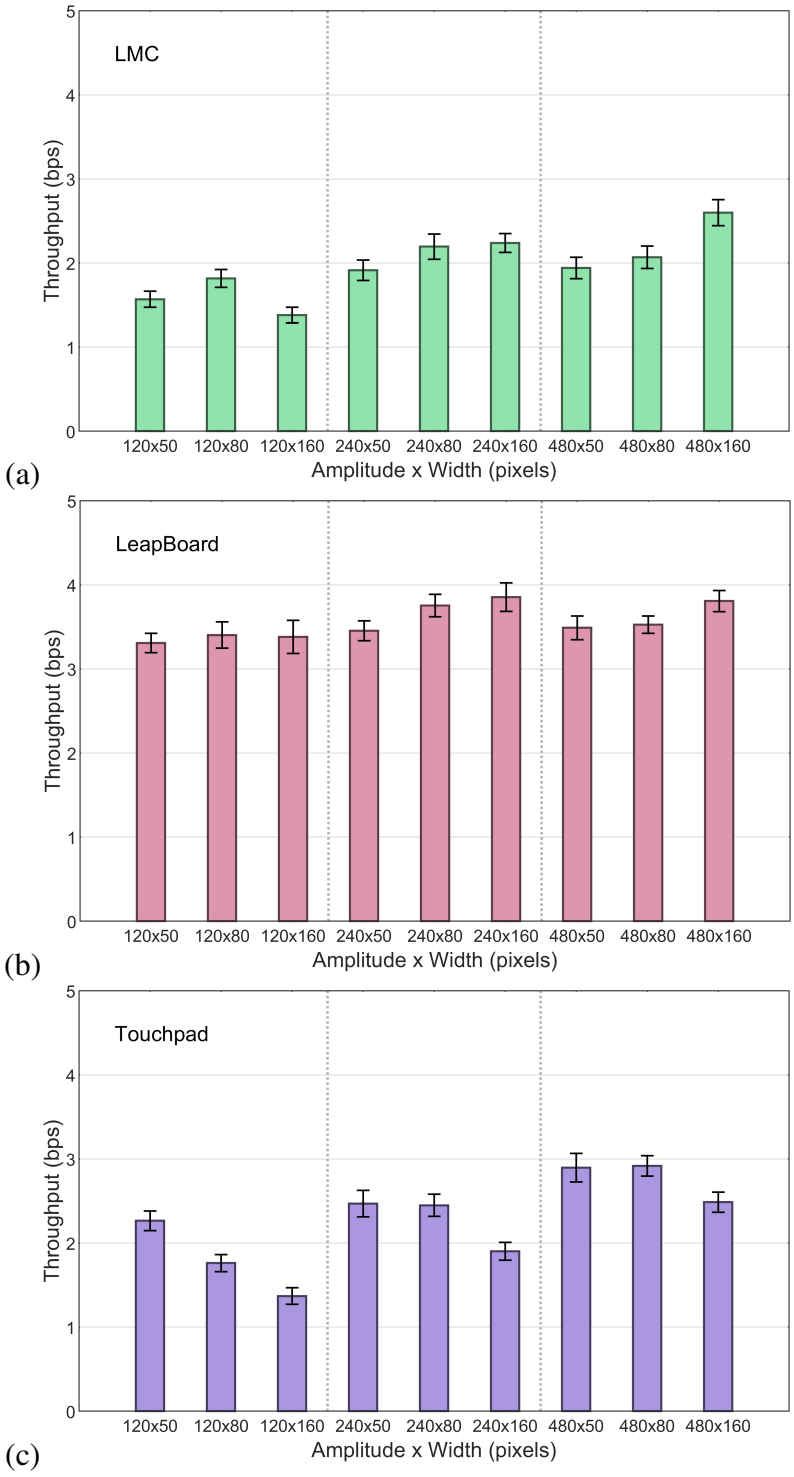


Figure 22: Throughput (bps) for the three selection methods by movement amplitude and target width. (a) LMC. (b) LeapBoard. (c) Touchpad. Error bars represent ±1 *SE*.

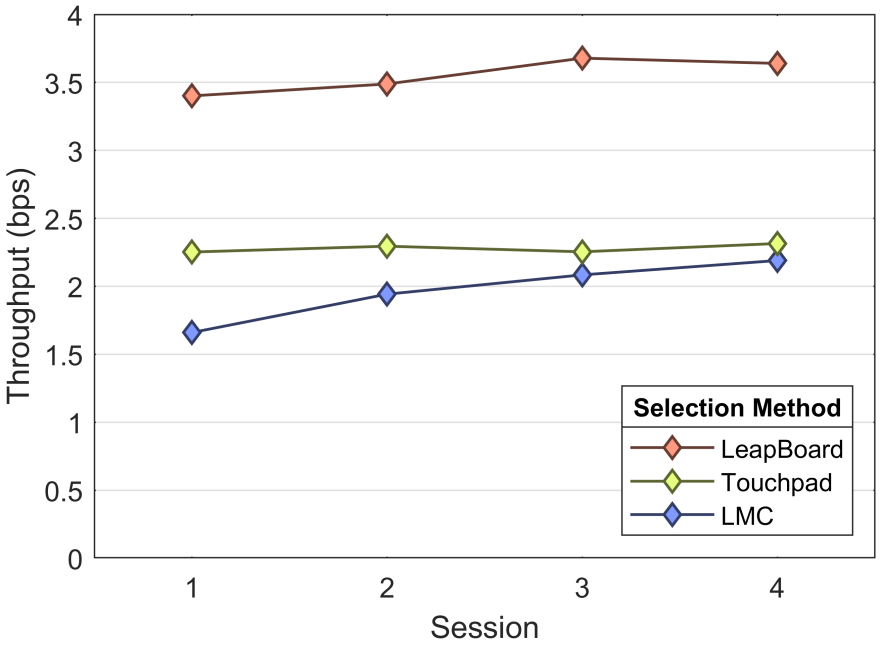


Figure 23: Throughput (bps) by session and selection method.

240 pixels.

The effect of session on *TP* was statistically significant ($F_{3,33} = 31.52$, $p < .0001$). Figure 23 shows the *TP* of the selection methods across the sessions. The effect of session on *TP* was statistically significant in LeapBoard ($F_{3,33} = 3.211$, $p < .05$) and LMC ($F_{3,33} = 7.297$, $p < .001$). The effect of session on the touchpad was not statistically significant.

TP of the touchpad and LMC was consistent with reported values by Seixas et al. [63]. Dube et al. [62] used amplitudes 80, 360, and 640 pixels with widths 25, 50, and 75. This resulted in a *TP* of 2.29 bps using the LMC with the tap gesture, which was higher than what was observed here using the LMC.

5.2.2 Error Rate

The grand mean for error rate was 6.9%. The breakdown by selection method is presented in Figure 24. For each box, $n = 432$ (12 participants × 4 Sessions × 3 Amplitudes × 3 Widths). Thus the outlier points represent a small proportion of the data. The box plots in the figure look unusual due to the high number of error-free sequences. Of the 432 trial sequences for each selection method, the number of error-free sequences was 151 (35.0%), 245 (56.7%), and 269 (62.3%) for LMC, LeapBoard, and the touchpad, respectively. Consequently, for the LeapBoard and touchpad conditions, the median error rate was 0%.

By selection method, the highest error rate was 12.8% for LMC, followed by 4.6% for Leap-

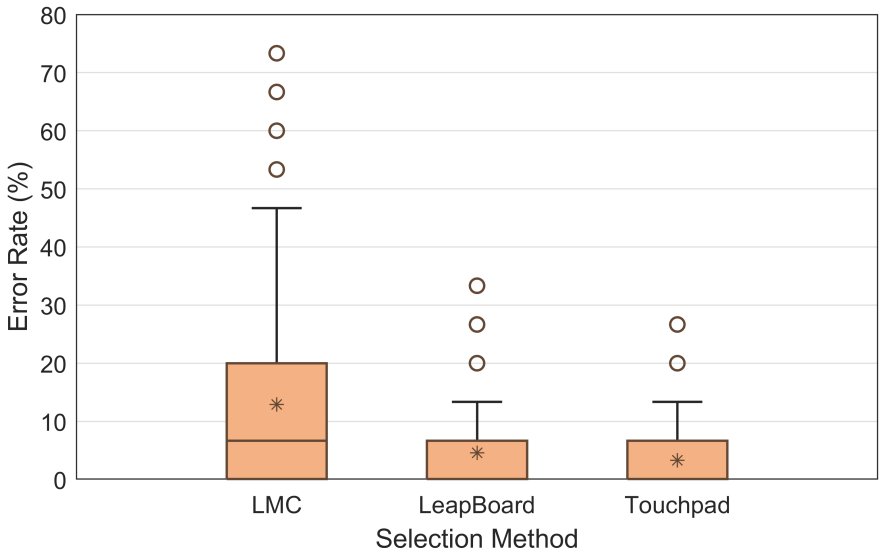


Figure 24: Error rate (%) by selection method.

Board, and 3.4% for the touchpad. The difference was statistically significant ($F_{2,22} = 23.13$, $p < .0001$). A Scheffé post hoc test showed that the LMC had a statistically significantly higher error rate than the LeapBoard and the touchpad, which could be attributed to the hand proximity to the device due to the setup. It was observed that using the mid-air tap gesture to select a target caused unintended hand movements, resulting in the cursor deviating from the target and increasing the error rate. The error rate was comparable between the LeapBoard and the touchpad, potentially indicating the effectiveness of eliminating the tap gesture for selection. Nevertheless, the LeapBoard exhibited a slightly higher error rate than the touchpad, which may be attributed to holding the positioning hand mid-air, making it susceptible to unintended movements.

An ANOVA showed the effect of amplitude on error rate was statistically significant ($F_{2,22} = 20.99$, $p < .0001$). A Scheffé post hoc test revealed a significant difference between amplitudes 120 and 480.

An ANOVA test showed the effect of Selection Method $\times$ Amplitude was statistically significant ($F_{8,88} = 22.33$, $p < .0001$). A Scheffé post hoc test showed that the LMC reported a statistically higher error rate than LeapBoard and touchpad for the amplitude of 480 pixels. Hand and LMC proximity issues led to poor gesture recognition in LMC sessions with a 480-pixel amplitude, especially for targets near the bottom of the screen. Participants often had to repeat the tap gesture, resulting in unintended hand movements and extra clicks. LeapBoard sessions, without the tap gesture, exhibited a significantly lower error rate.

An ANOVA showed that the effect of width on error rate was statistically significant ($F_{2,22} = 70.8$, $p < .0001$). A Scheffé post hoc test revealed a significant difference for width 50 with the other two widths. It's important to highlight that the experiment initially began with a width of 40

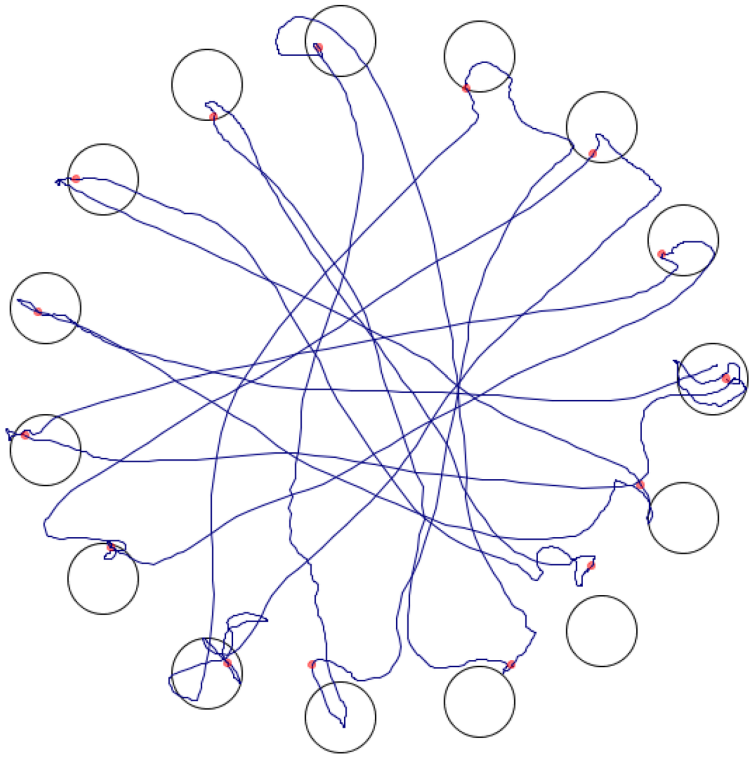


Figure 25: Trace example for LMC and $W = 50$ pixels. There were difficulties selecting targets near the bottom of the layout circle due to the close proximity of the user's hand to the Leap Motion Controller.

pixels instead of 50 pixels, and this configuration was found highly error-prone and challenging for selection. Consequently, the width was adjusted from 40 to 50 pixels.

An ANOVA revealed that the Selection Method $\times$ Width interaction effect on error rate was statistically significant ($F_{8,88} = 34.32$, $p < .0001$). The LMC with width 50 pixels had a statistically significant difference with all other widths on LMC, LeapBoard, and touchpad. LMC was also statistically significantly different than the touchpad for the width of 80 pixels.

The Selection Method $\times$ Amplitude $\times$ Width interaction was found to be statistically significant ($F_{26,286} = 26.23$, $p < .0001$). A Scheffé post hoc test revealed that the LMC with an amplitude of 480 pixels and width of 50 pixels significantly differed from all amplitude-width interactions on the other selection methods. Additionally, LMC with an amplitude of 240 pixels and a width of 50 pixels was statistically different than all amplitude-width combinations on the touchpad. It was observed throughout the experiment that selecting the smallest targets ($W = 50$ pixels) closer to the bottom of the screen was frustrating to the participants when using the LMC due to gesture recognition difficulties near the LMC. See Figure 25 for an example. This was not the case when using the LeapBoard or touchpad.

Figure 26 shows the error rate of the selection methods across the sessions. The effect of the

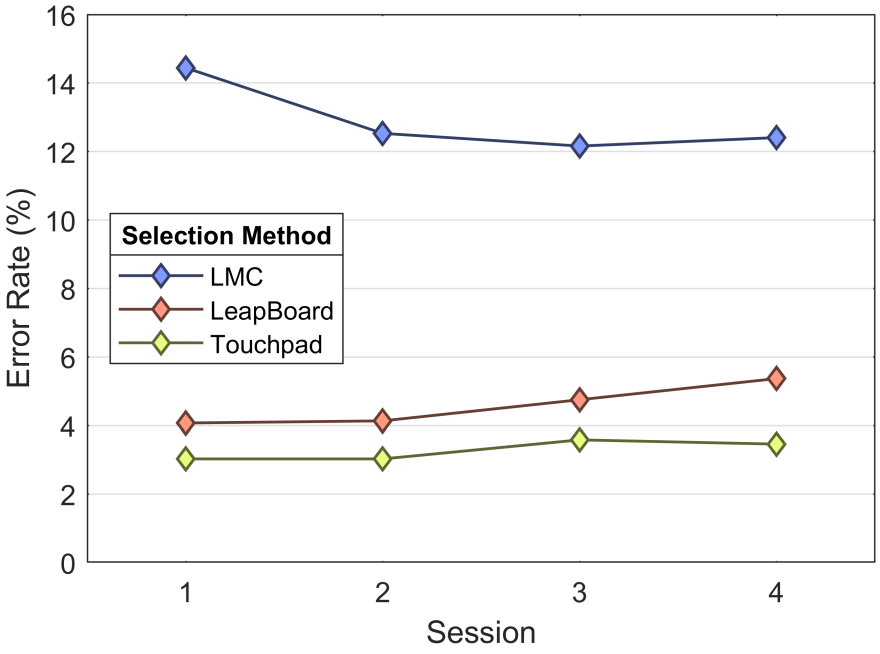


Figure 26: Error rate (%) by selection method. Asterisks show the mean of the responses.

session on the error rate was not statistically significant in any of the selection methods, as the errors were mostly attributed to gesture recognition problems and were not solved by practice.

The literature review showed that the mid-air gestures detected by LMC consistently exhibited markedly higher error rates than conventional input devices such as mouse and touchpads. This discerning observation remained evident in our study; however, a significant improvement was achieved with LeapBoard.

Seixax et al. [63] observed the error rate for the LMC using the screen tap gesture for selection was about three times larger than the error rate for the touchpad, which was similar to what was observed in this study. Dube et al. [62] reported an error rate of 1.77% for the tap gesture. This is lower than all the error rates of all selection methods in this study. Although their experiment used higher amplitudes and lower widths, the error rate was lower, possibly due to the LMC position and distance to the participant's hand, which was higher than what was used in this study.

5.2.3 Hand Fatigue and Preference

In the last session, participants were asked to rank their overall level of hand fatigue for the selection method; refer to the Appendix. Responses were recorded on a 10-point scale from 1 (*no fatigue*) to 10 (*very fatiguing*). The means were 6.4, 1.8, and 2.0 for the LMC, LeapBoard, and touchpad, respectively, indicating somewhat high hand fatigue for the LMC and similar low hand fatigue for the LeapBoard and touchpad. A Friedman test indicated a significant difference be-

tween the selection methods on hand fatigue (χ^2 = 16.17, $p < .001$, df = 2). Pairwise comparisons using Conover's F indicated that the LMC rating for hand fatigue was significantly higher than for Leapboard or touchpad (p ¡ .05). However, there was no significant difference between Leapboard and touchpad.

Participants were also asked to rank the selection methods based on their overall preference. Half the participants (6) ranked LeapBoard first, half ranked the touchpad first. All participants indicated that LMC was their least preferred selection method.

The feedback from participants using the LMC indicated a generally positive experience with comfortable movement to the target (pointing to the target). The natural mapping of hand movement to cursor movement was perceived as "entertaining," but most participants found the selection difficult, as this study hypothesized. Challenges such as the need for a tense hand position for target selection and difficulties with lower widths and higher amplitudes were noted as downfalls in selecting the targets.

For the LeapBoard, users reported feeling less accurate and facing challenges with high amplitudes. Fatigue from keeping the arm elevated was mentioned, but the system's speed was recognized as a positive aspect. Lower amplitudes and higher widths were deemed easier to select. Regarding the touchpad, participants noted challenges with cursor movement, especially for higher amplitudes, while acknowledging its speed and accuracy. Preferences varied, with some participants stating they were used to it, while others found it slower but more accurate than the LeapBoard.

6 Third Experiment - Comparing Fun and Performance: A User Study on Children's Gaming Experiences with Mid-Air Hand Gestures [22]

6.1 Methodology

This section outlines the assessment of the performance and fun in a game using the input methods described in section 2.4.

6.1.1 Participants

Eighteen children (nine girls and nine boys) aged 5 to 7 were recruited from local McDonald's and Ikea Foodcourt to participate in this user study with their parent's permission. The children's parents were asked to sign a consent form before the experiment begins. They received $20 after finishing the experiment.

6.1.2 Apparatus

The Animal Match-up game ran on an HP Envy x360 laptop with a screen resolution of 1536×864 operating with Windows 11. An Intel RealSense D435 camera facilitated gesture capture. See Figure 27. A custom Python program was employed, leveraging the MediaPipe v0.7 library to detect hand gestures.

6.1.3 Procedure

The experiment was conducted in local public places such as McDonald's and Ikea Foodcourt. Ethical approval for this study was obtained from the Office of Research Ethics at York University. The researcher approached the parents of children in these places and explained the experiment to them. Interested parents then discussed the experiment with their children, and those children who showed interest were recruited to participate in the study on the same day and at the same location. Tables in quieter areas of these places were used as the experiment area. The participants were tested individually. Before the experiment, the children's parents were informed and asked to sign a consent form on behalf of the children.

The researcher welcomed the children and invited them to sit comfortably in front of a laptop equipped with a RealSense camera. They then explained the game and provided a demonstration.

[22] This work will be published in the *British HCI* conference in July 2024.

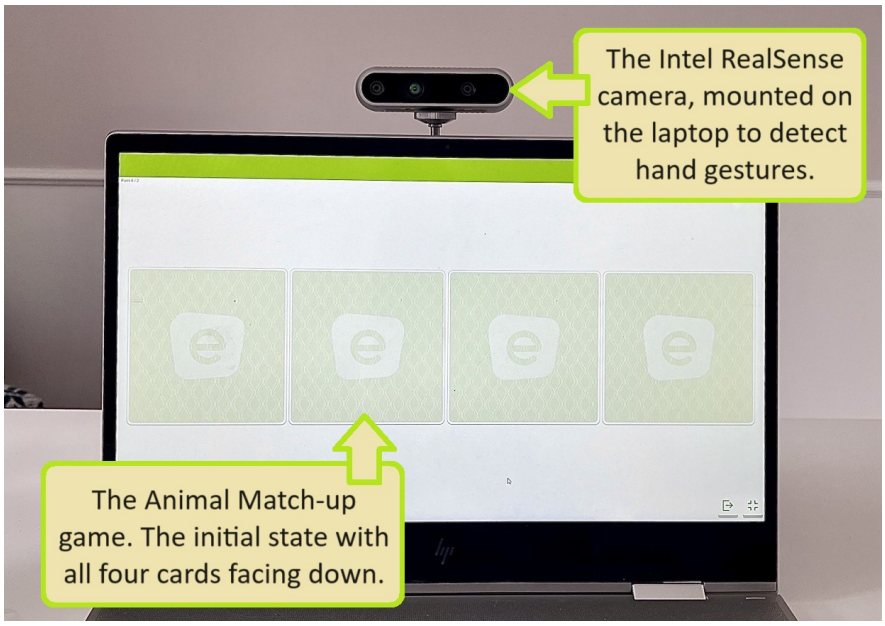


Figure 27: The Animal Match-up game running on the laptop with the Intel RealSense camera mounted on it.

The children were informed that they would use three different input methods during the experiment. Before starting the trials with each input method, the researcher described and demonstrated it to the children. To familiarize themselves, the children completed three practice trials with each new input method. Throughout the experiment, the children were allowed to take breaks whenever needed.

Each participant performed five trials using each input method. Figure 28 shows the experiment setting.

To counterbalance the order effects, the participants were divided into six groups of three, each performing the experiment in a unique order.

Each trial began with four cards appearing on the laptop display. See Figure 27. Timing began at the beginning of each trial and stopped when the two pairs in each trial were matched. The time to complete each trial as well as the number of selected cards to finish the trial were recorded.

The researcher took notes of participants' expressed emotions throughout the experiment to record information on their feelings (e.g., frustration, excitement, etc.). After finishing the experiment, the Fun Sorter and the Again Again Table from the Fun Toolkit were used to gather information about children's fun and preferences. In the Fun Sorter activity, the children were asked to sort the input methods based on ease of use and likability. See Figure 29a. This activity involved placing cards with the image of the input method on the table slots. In the Again Again table, the children were asked to answer yes, no, or maybe to whether they would like to play the

Figure 28: A participant playing the Animal Match-up game using three input methods: (a) Mouse, (b) Air Mouse, and (c) Finger Numbers.

game again using each input method. See Figure 29b.

6.1.4 Design

This experiment used a 3 × 5 within-subjects design with the following independent variables and levels:

- Input Method (Air Mouse, Finger Numbers, and Mouse)
- Trial (1, 2, 3, 4, 5)

The dependent variables were trial completion time (s) and the number of selected cards to finish each trial. Qualitative data was obtained using the Fun Toolkit as explained in section 6.1.3.

The total number of trials per participant was 15 (3 input methods × 5 trials) each requiring a minimum of four selections. The total testing time for each participant was approximately 15 minutes.

6.2 Results and Discussion

6.2.1 Trial Completion Time

The grand mean for trial completion time was 20.0 s per trial. By input method, the means were 13.3 s for the mouse, 23.9 s for the Air Mouse, and 22.6 s for Finger Numbers. See Figure 30. An ANOVA test showed that the effect of the input method on trial completion time was statistically significant ($F_{2,34} = 25.02$, $p < .0001$). Scheffé post hoc tests showed a significant difference between the mouse and the other two input methods.

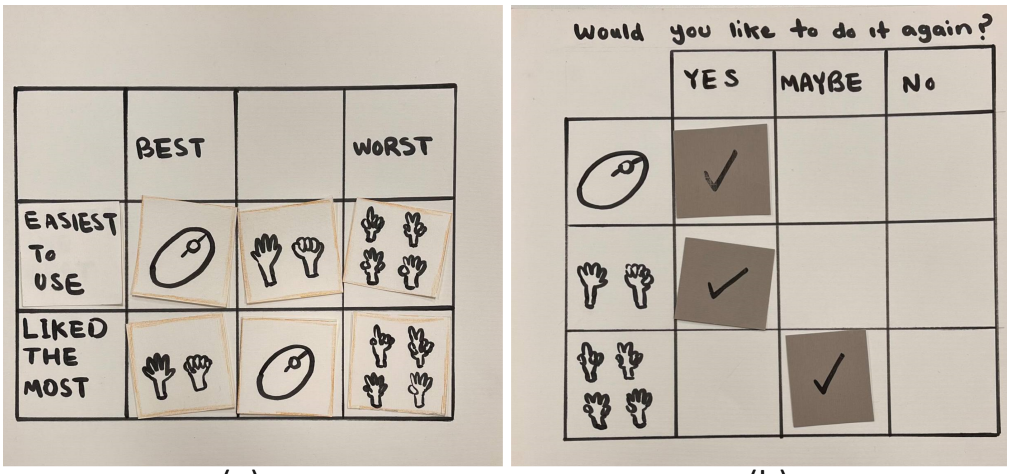


Figure 29: The Fun Toolkit tools used in this study: (a) The ease of use Fun Sorter on the top row and the likability Fun Sorter on the bottom row (filled randomly). (b) The Again Again table.

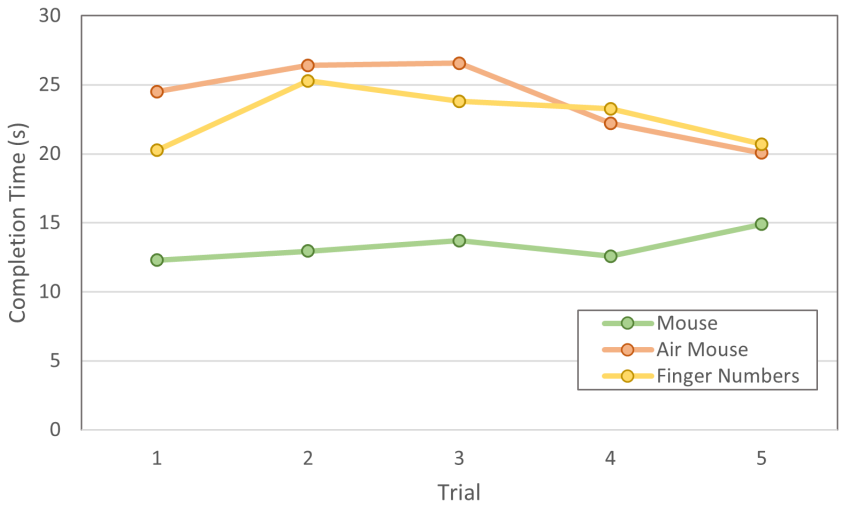


Figure 30: The average trial completion time (s) using the three input methods.

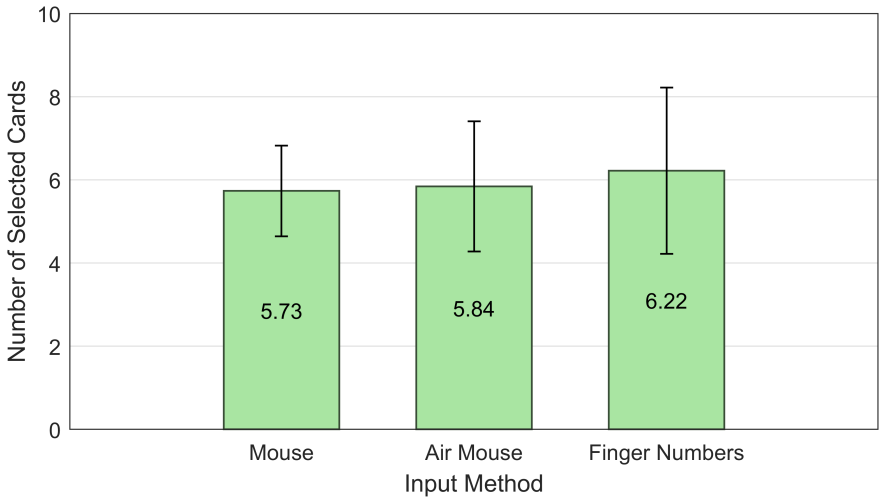


Figure 31: Number of selected cards per trial by input method. Error bars show ± 1 *SD*.

The effect of the trial on trial completion time was not statistically significant ($F_{4,68} = 2.33$, $p > .05$). Similarly, the Input Method $\times$ Trial interaction effect was found to be not statistically significant ($F_{8,136} = 1.28$, $p > .05$).

6.2.2 Number of Selected Cards

The grand mean for number of selected cards per trial was 5.93. This metric relates to accuracy, with 4 (the number of cards) as a baseline for a perfect trial. The number rises above 4 in correspondence with cards incorrectly selected during a trial. By input method, the means were 5.73 for the mouse, 5.84 for Air Mouse, and 6.22 for Finger Numbers. See Figure 31. The effect of input method on the number of selected cards was not statistically significant ($F_{2,34} = 2.21$, $p > .05$). Similarly, the effect of the trial on the number of selected cards was not statistically significant ($F_{4,68} = 0.48$, ns), nor was the Input Method $\times$ Trial interaction effect ($F_{8,136} = 0.49$, ns).

6.2.3 Fun and Preference

Figure 32 shows the results from the ease of use Fun Sorter. For the analysis, the ease of use Fun Sorter results were converted to values 1, 2, and 3, with 1 being the easiest to use and 3 hardest to use. The mean, median, mode, and standard deviation (*SD*) were calculated to analyze the perceived ease of use. For the mouse, the mean score was 1.44, indicating that, on average, it was rated relatively easy to use. The median and mode were 1, suggesting that most children ranked the mouse as the easiest to use ($SD = 0.7$). For the Finger Numbers, the mean score was 2.22, indicating a perceived difficulty between the mouse and the Air Mouse. The median and mode were 2, showing that the Finger Numbers was most commonly ranked moderately easy to

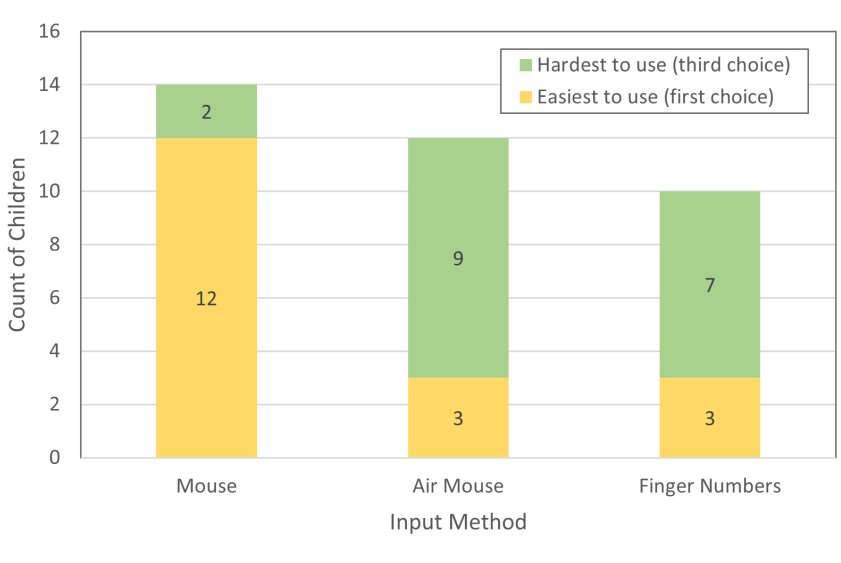


Figure 32: Results from the ease of use Fun Sorter: Number of children who selected each input method as easiest or hardest to use.

use (SD = 0.73). The Air Mouse had a mean score of 2.33, indicating a somewhat higher perceived difficulty than the other two input methods. The median was 2.5, and the mode was 3, suggesting that the Air Mouse was mainly ranked as the hardest to use (SD = 0.76). A Friedman test showed a significant difference in the ease of use of the input methods (χ^2 = 8.44, $p < .05$, df =2) between the mouse and the mid-air gesture-based input methods.

Figure 33 shows the results from the likability Fun Sorter. Similarly to the ease of use Fun Sorter, the results from the likability Fun Sorter were converted to values 1, 2, and 3, with 1 being liked the most and 3 being liked the least. The mean, median, mode, and SD were calculated to analyze the likability of each input method. The mean, median, mode, and SD were calculated to analyze the likability of each input method. The mean scores were 1.5, 2.05, and 2.44 for the mouse, Finger Numbers, and the Air Mouse, respectively. The median and mode were 1 (SD = 0.7) for the mouse, 2 (SD = 0.64) for the Finger Numbers, and 3 for the Air Mouse (SD = 0.85). A Friedman test showed a significant difference in the likability of the input methods (χ^2 = 8.11, $p <$ .05, df =2) between the mouse and the Air Mouse.

Among the 18 participants, six had no prior experience with a mouse. This group struggled with clicking the mouse during the practice trials, and their hand position on the mouse had to be corrected. Some used the scrolling wheel to click, which was corrected before the experiment trials. Interestingly, while two participants within this group selected the mouse as the easiest input method, none chose it as their most liked input method. Three participants chose the Air Mouse, and three chose Finger Numbers as their most liked input method. When asked what they liked the least, two participants within this group chose the mouse, three chose the Air Mouse, and one

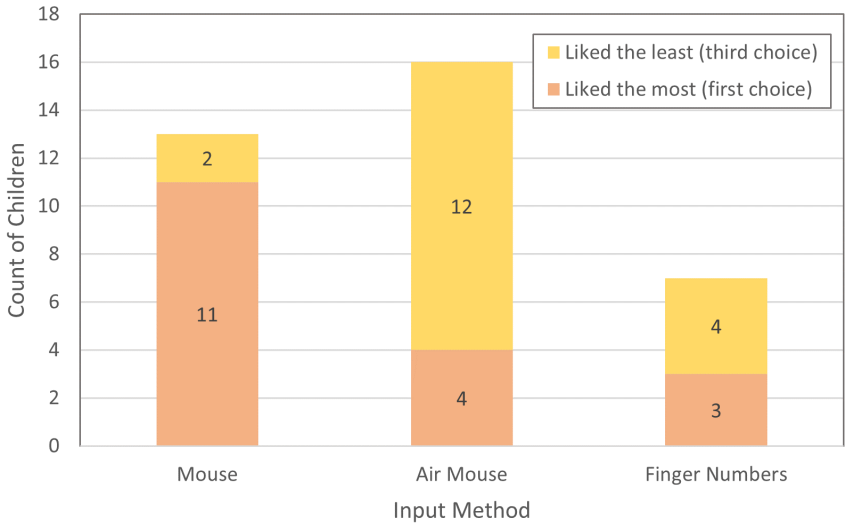


Figure 33: Results from the likability Fun Sorter: Number of children who selected each input method as their most or least liked input method.

selected Finger Numbers.

Overall, seven participants didn't select the mouse as their most liked input method; six had no previous experience using the mouse. Similarly, a total of six participants, four of whom had no experience using the mouse before, did not select the mouse as the easiest to use.

Figure 34 shows the results from the Again Again Table, in which the children answered if they would like to play the game again using each input method. The mouse had the highest number of *yes* responses. Three participants, two of whom had no prior experience with the mouse, responded *maybe*, which could indicate the role of familiarity in the high number of positive responses.

During the Air Mouse trials, children encountered problems with gesture detection, perhaps due to issues with detecting small hands. Sometimes, the fist gesture that triggered a selection was not detected quickly, and children had to repeat the gesture multiple times. Due to this, some children got frustrated and moved their hands vigorously toward the camera in the fist position to make the camera detect it. At times, the children forgot to open their fists after making a selection. This led to unintended multiple selections of the same card. Additionally, despite moving their hands, the cursor didn't move while they maintained the fist gesture. They needed to be reminded to open their fists.

Similarly, there were instances where children remembered to open their fists, but only partially, resulting in a half-fist gesture instead of the fully open hand gesture needed to move the cursor. Consequently, the cursor remained stationary on the screen despite the children moving their hands. To address this issue, they often brought their hand closer to the camera for detection, occasionally getting too close for the camera to detect anything, leading to frustration. Their pos-

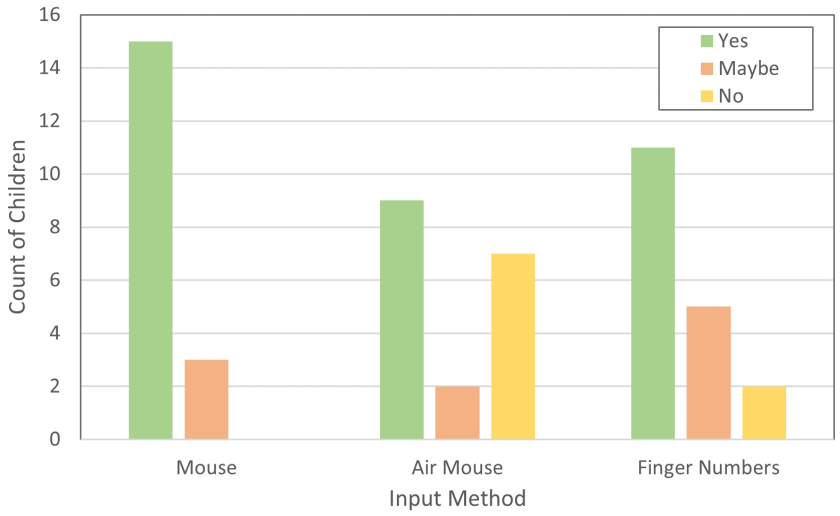


Figure 34: Responses from the Again Again table for the question, ”Would you like to do it again?”

ture had to be corrected by the researcher, and they had to be reminded to open their fists fully after each selection. It was also observed that some children moved their bodies and leaned on their sides when moving their hands in front of the camera. Consequently, the Air Mouse encountered more mixed reactions, with seven children responding that they did not want to play the game using it again and two showing uncertainty.

Finger Numbers received more *yes* responses and fewer *no* responses than the Air Mouse. However, there were also a notable number of *maybe* responses (five), suggesting some uncertainty or reservation about this input method. In the Finger Number trials, the gestures were sometimes not detected quickly. Like the Air Mouse trials, the children moved their hands toward the camera to make it detect the gesture, and they seemed frustrated. Sometimes, they were misidentified, and a card the child did not intend to select was selected, leading to confusion. Some children verbalized their feelings in these instances with sentences such as ”I didn’t wanna do that.”

Sometimes, the gestures were not performed correctly; for example, they showed their thumb instead of the gesture for selecting the first card, or they held up their middle, index, and pinky fingers instead of the gesture for selecting the third card. This showed that sometimes the children relied on counting fingers up instead of the gestures taught to them. In many instances, the children took time to think about the next gesture they wanted to perform.

Most children had to look at their hands while thinking of the next gesture. Some did that by bringing their hand down, outside the camera range; some children held their hands in front of the camera while thinking and opening and closing their fingers. This resulted in wrong and unintentional selections. Some children used both hands interchangeably, making one gesture with one hand, bringing it down, and using the other for the next gesture.

7 Conclusion

7.1 Findings from the First Experiment

In a five-session longitudinal evaluation of H4VR, participants reached an average entry speed of 3.63 and 4.57 words per minute on the cross and flat keyboards, respectively. As the sessions progressed, participants reported a significantly higher entry speed with an improvement of 65.7% and 93.6% on the cross and flat keyboards, respectively. The flat keyboard had a statistically significant higher entry speed than the cross keyboard.

The average error rate for both keyboards was low; however, the flat keyboard had a significantly higher error rate than the cross. More sessions of using the H4VR keyboards can lead to higher entry speed.

7.2 Findings from the Second Experiment

Our investigation's hypothesis was based on the idea that depending solely on mid-air gesture-based interactions, such as those involving Leap Motion Controller, could have limitations and may not perform as effectively as traditional methods like touchpads. One possible explanation for this observed discrepancy is related to the process of target selection. LeapBoard utilizes mid-air hand movement for moving the cursor on the screen and replaces mid-air gestures for target selection with physical keys on a keyboard to make up for that shortcoming. In the user study, LeapBoard outperformed the touchpad with 57.1% higher *TP* and surpassed LMC by 80.2%. The differences were statistically significant, with a Scheffé post hoc test highlighting the superiority of LeapBoard over the other two methods. When examining *TP* across different amplitudes, LeapBoard consistently outperformed LMC and touchpad. Although it is a two-handed method, the primary investigation was based on removing the selection gestures from the pointing hand, and that makes the resulting method a suitable alternative to methods such as touchpads. The *TP* was highest for LeapBoard at 7.58 bps for an amplitude of 120 pixels. Similarly, the analysis of *TP* across widths indicated that LeapBoard excelled, particularly at 160 pixels, with a statistically significant difference compared to the LMC and touchpad. LeapBoard resulted in a 73.6% smaller error rate than LMC and 25.9% smaller than the touchpad.

A primary takeaway message from this work is that mid-air gestural input methods may benefit by off-loading selection to a separate hand, perhaps using a key on a keyboard, or to some other channel of input. Fatigue issues must also be considered if interaction is expect over a prolonged period of time. The LeapBoard device presented herein offers a particular solution where mid-air gestural input might be combined with conventional use of a computer keyboard.

7.3 Findings from the Third Experiment

The mouse had a significantly lower trial completion time than the gesture-based input methods. Finger Numbers had a 5.75% lower trial completion time than the Air Mouse. It is important to note that Finger Numbers used four gestures while the Air Mouse relied on only two. The Finger Numbers had the highest number of cards selected to finish the trials. This indicates the high occurrences of unintended or misclassified gestures with this input method. The mouse was statistically significantly perceived as the easiest and most liked input method. The Finger Numbers was slightly more liked (19.02%) than the Air Mouse and was perceived almost as easy to use as the Air Mouse.

However, it is notable that all participants without prior experience using a mouse chose one of the two mid-air gesture-based input methods as their most liked input method. The mid-air gesture-based input methods underwent gesture detection delays, sometimes frustrating and confusing the children. Learning the gestures was also a part that was more challenging to the participants than having to use a mouse, even for the ones who had no prior experience using the mouse.

However, the analysis of willingness to use the input methods again showed no significant difference among the three input methods, suggesting that despite challenges, participants were generally open to retrying the mid-air gesture-based input methods. Overall, while the mouse demonstrated the best performance, fun, and user preference, the study highlights the importance of considering user familiarity and usability challenges associated with mid-air gesture-based input methods, like Air Mouse and Finger Numbers, among younger users.

Further refinements in gesture detection for smaller hands, training models specifically for gesture detection for children, and more extended user training may be necessary to improve the usability and acceptance of these alternative input methods in interactive systems designed for children.

7.4 Future Work

Throughout the experiments, areas for potential improvement were identified through careful observation and analysis. When gestures are used, gesture detection plays an integral role, and it was a common observation that the accuracy of gesture detections and classifications could be improved.

In the first user study, conducted in VR using an HMD, the hand-tracking ability of the HMD can replace the external camera. Hand-tracking, a feature that enables VR users to use their hands as input devices rather than controllers, tracks the users' hands and detects hand gestures. Some HMDs, such as HoloLens 2, the Leap Motion, and the Oculus Quest, offer hand-tracking using built-in cameras and sensors on the HMD. The user is also given a representation of their hand in the virtual environment. Using the HMD hand-tracking can reduce hand fatigue since the hand

does not need to be up and facing a camera. The gestures can be made anywhere with any hand orientation as long as it is recognizable by the HMD.

During the experiment, it was observed that gesture recognition could be incorrect if the hand was not fully facing the camera. This issue can also be resolved with HMD hand-tracking, leading to lower mistakes and higher entry speed. Having a hand representation in a virtual environment can also improve the user experience, which again can be achieved using the HMD hand-tracking ability.

Some participants also noted that fast double selection on the same key only gets recognized as one selection. Hardware that allows more immediate gesture recognition, another future improvement, can address this issue and improve entry speed.

In the second user study, a persistent issue with gesture recognition was when the hand was too close to the LMC, i.e., for selecting the targets at the lower parts of the screen. This issue needs to be addressed in future studies. A design that positions the LMC at a higher distance from the hand could solve this problem. Replacing the LMC with other gesture detection technology that can detect gestures in close proximity can address this issue.

Lastly, by integrating machine learning models, future research can increase the accuracy of gesture detections and lower the error rates. Especially when designing systems for children who have smaller hands and the built-in SDKs might not perform as accurately.

References

[1] P. Koutsabasis and P. Vogiatzidakis, "Empirical research in mid-air interaction: A systematic review," *International Journal of Human–Computer Interaction*, vol. 35, no. 18, pp. 1747–1768, 2019.

[2] R. A. Bolt, ""put-that-there" voice and gesture at the graphics interface," in *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques – SIGGRAPH '80*, (New York), pp. 262–270, ACM, 1980.

[3] S. A. Grandhi, G. Joue, and I. Mittelberg, "Understanding naturalness and intuitiveness in gesture production: Insights for touchless gestural interfaces," in *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems – CHI '11*, (New York), pp. 821–824, ACM, 2011.

[4] S. Pei, A. Chen, J. Lee, and Y. Zhang, "Hand interfaces: Using hands to imitate objects in ar/vr for expressive interactions," in *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems – CHI '22*, (New York), pp. 1–16, ACM, 2022.

[5] H.-S. Hsiao and J.-C. Chen, "Using a gesture interactive game-based learning approach to improve preschool children's learning performance and motor skills," *Computers & Education*, vol. 95, pp. 151–162, 2016.

[6] S. Nitnaware and A. Bagde, "Hand gesture recognition to facilitate tasks for the disabled," in *The Second International Conference on Artificial Intelligence and Smart Energy – ICAIS*, (New York), pp. 949–953, IEEE, 2022.

[7] S. Vidal and G. Lefebvre, "Gesture based interaction for visually-impaired people," in *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries – NordiCHI '10*, (New York), pp. 809–812, ACM, 2010.

[8] S. Ahmed, K. D. Kallu, S. Ahmed, and S. H. Cho, "Hand gestures recognition using radar sensors for human-computer-interaction: A review," *Remote Sensing*, vol. 13, no. 3, p. 527, 2021.

[9] K. A. Bhaskaran, A. G. Nair, K. D. Ram, K. Ananthanarayanan, and H. N. Vardhan, "Smart gloves for hand gesture recognition: Sign language to speech conversion system," in *2016 International Conference on Robotics and Automation for Humanitarian Applications (RAHA)*, (New York), pp. 1–6, IEEE, 2016.

[10] K. Suri and R. Gupta, "Classification of hand gestures from wearable imus using deep neural network," in *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, (New York), pp. 45–50, IEEE, 2018.

[11] V. Moysiadis, D. Katikaridis, L. Benos, P. Busato, A. Anagnostis, D. Kateris, S. Pearson, and D. Bochtis, "An integrated real-time hand gesture recognition framework for human–robot interaction in agriculture," *Applied Sciences*, vol. 12, no. 16, p. 8160, 2022.

[12] A. Withana, T. Kaluarachchi, C. Singhabahu, S. Ransiri, Y. Shi, and S. Nanayakkara, "Wavesense: Low power voxel-tracking technique for resource limited devices," in *Proceedings of the Augmented Humans International Conference*, (New York), pp. 1–7, ACM, 2020.

[13] J. Kim, S. Mastnik, and E. André, "Emg-based hand gesture recognition for realtime biosignal interfacing," in *Proceedings of the 13th International Conference on Intelligent User Interfaces – IUI '08*, (New York), pp. 30–39, ACM, 2008.

[14] G. Luo, P. Yang, M. Chen, and P. Li, "Hci on the table: robust gesture recognition using acoustic sensing in your hand," *IEEE Access*, vol. 8, pp. 31481–31498, 2020.

[15] J. Li, J. Zhong, and N. Wang, "A multimodal human-robot sign language interaction framework applied in social robots," *Frontiers in Neuroscience*, vol. 17, p. 1168888, 2023.

[16] M. W. Krueger, T. Gionfriddo, and K. Hinrichsen, "Videoplace—an artificial reality," in *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems - CHI '85*, (New York), pp. 35–40, ACM, 1985.

[17] M. Speicher, A. M. Feit, P. Ziegler, and A. Krüger, "Selection-based text entry in virtual reality," in *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems - CHI '18*, (New York), pp. 1–13, ACM, 2018.

[18] D. Yu, K. Fan, H. Zhang, D. Monteiro, W. Xu, and H.-N. Liang, "Pizzatext: Text entry for virtual reality systems using dual thumbsticks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 11, pp. 2927–2935, 2018.

[19] K.-J. Wang, Q. Liu, Y. Zhao, C. Y. Zheng, S. Vhasure, Q. Liu, P. Thakur, M. Sun, and Z.-H. Mao, "Intelligent wearable virtual reality (vr) gaming controller for people with motor disabilities," in *IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, pp. 161–164, 2018.

[20] M. Oudah, A. Al-Naji, and J. Chahl, "Hand gesture recognition based on computer vision: A review of techniques," *Journal of Imaging*, vol. 6, no. 8, 2020.

[21] W. Xu, H.-N. Liang, A. He, and Z. Wang, "Pointing and selection methods for text entry in augmented reality head mounted displays," in *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, (New York), pp. 279–288, IEEE, 2019.

[22] C. Yildirim, "Point and select: Effects of multimodal feedback on text entry performance in virtual reality," *International Journal of Human-Computer Interaction*, vol. 0, no. 0, pp. 1–15, 2022.

[23] S. Chen, J. Wang, S. Guerra, N. Mittal, and S. Prakkamakul, "Exploring word-gesture text entry techniques in virtual reality," in *Extended Abstracts of the ACM SIGCHI Conference on Human Factors in Computing Systems - CHi '19*, (New York), pp. 1–6, ACM, 2019.

[24] C. Lim, J. Kim, and M. J. Kim, "Thumble: One-handed 3d object manipulation using a thimble-shaped wearable device in virtual reality," in *Adjunct Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology - UIST '22*, (New York), pp. 17.1–17.3, ACM, 2022.

[25] M. K. Bhuyan, D. Ajay Kumar, K. F. MacDorman, and Y. Iwahori, "A novel set of features for continuous hand gesture recognition," *Journal on Multimodal User Interfaces*, vol. 8, no. 4, pp. 333–343, 2014.

[26] J. I. Koh, J. Cherian, P. Taele, and T. Hammond, "Developing a hand gesture recognition system for mapping symbolic hand gestures to analogous emojis in computer-mediated communication," *ACM Transactions on Interactive Intelligent Systems*, vol. 9, no. 1, pp. 1–35, 2019.

[27] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[28] I. S. MacKenzie, R. W. Soukoreff, and J. Helga, "1 thumb, 4 buttons, 20 words per minute: Design and evaluation of h4-writer," in *Proceedings of the 2011 ACM Symposium on User Interface Software and Technology - UIST '11*, (New York), ACM, 2011.

[29] B. Bajer, I. S. MacKenzie, and M. Baljko, "Huffman base-4 text entry glove (H4 TEG)," in *Proceedings of the 16th International Symposium on Wearable Computers*, (New York), pp. 41–47, IEEE, 2012.

[30] I. S. MacKenzie and S. J. Castellucci, "Eye on the message: Reducing attention demand for touch-based text entry," *International Journal of Virtual Worlds and Human-Computer Interaction (VWHCI)*, vol. 1, no. 1, pp. 1–9, 2013.

[31] M. Ponti, "Screen time and preschool children: Promoting health and development in a digital world," *Paediatrics & Child Health*, vol. 28, pp. 184–192, 05 2023.

[32] D. A. Lieberman, M. C. Fisk, and E. Biely, "Digital games for young children ages three to six: From research to design," *Computers in the Schools*, vol. 26, no. 4, pp. 299–313, 2009.

[33] N. Iten and D. Petko, "Learning with serious games: Is fun playing the game a predictor of learning success?," *British Journal of Educational Technology*, vol. 47, no. 1, pp. 151–163, 2016.

[34] J. Long, "Just for fun: Using programming games in software programming training and education," *Journal of Information Technology Education: Research*, vol. 6, no. 1, pp. 279–290, 2007.

[35] D. R. A. Rambli, W. Matcha, and S. Sulaiman, "Fun learning with ar alphabet book for preschool children," *Procedia Computer Science*, vol. 25, pp. 211–219, 2013.

[36] L. C. Vieira and F. S. C. da Silva, "Assessment of fun in interactive systems: A survey," *Cognitive Systems Research*, vol. 41, pp. 130–143, 2017.

[37] G. Sim, S. MacFarlane, and J. Read, "All work and no play: Measuring fun, usability, and learning in software for children," *Computers & Education*, vol. 46, no. 3, pp. 235–248, 2006. Virtual Learning?

[38] S. C. Chan, C. J. Wan, and S. Ko, "Interactivity, active collaborative learning, and learning performance: The moderating role of perceived fun by using personal response systems," *The International Journal of Management Education*, vol. 17, no. 1, pp. 94–102, 2019.

[39] S. Elton-Chalcraft and K. Mills, "Measuring challenge, fun and sterility on a 'phunometre' scale: Evaluating creative teaching and learning with children and their student teachers in the primary school," *Education 3-13*, vol. 43, no. 5, pp. 482–497, 2015.

[40] D. Lucardie, "The impact of fun and enjoyment on adult's learning," *Procedia-Social and Behavioral Sciences*, vol. 142, pp. 439–446, 2014.

[41] J. Willis, "The neuroscience of joyful education," *Educational leadership*, vol. 64, no. 9, pp. 1–5, 2007.

[42] J. C. Read, "Validating the fun toolkit: an instrument for measuring children's opinions of technology," *Cognition, Technology & Work*, vol. 10, pp. 119–128, 2008.

[43] J. C. Read and S. MacFarlane, "Using the Fun Toolkit and other survey methods to gather opinions in child computer interaction," in *Proceedings of the 2006 Conference on Interaction Design and Children - IDC '06*, (New York), pp. 81–88, ACM, 2006.

[44] I. S. MacKenzie, "KSPC (keystrokes per character) as a characteristic of text entry techniques," in *Proceedings of the Fourth International Symposium on Human-Computer Interaction with Mobile Devices – MobileHCI '02*, (Berlin), pp. 195–210, Springer, 2002.

[45] I. S. MacKenzie, *Human-computer interaction: An empirical research perspective*. Cambridge, MA: Morgan Kaufmann (an imprint of Elsevier), 2nd ed., 2024.

[46] I. S. MacKenzie, "Fitts' law," in *Handbook of human-computer interaction* (K. L. Norman and J. Kirakowski, eds.), pp. 349–370, Hoboken, NJ: Wiley, 2018.

[47] I. S. MacKenzie and W. Buxton, "Extending fitts' law to two-dimensional tasks," in *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems – CHI '92*, (New York), pp. 219–226, ACM, 1992.

[48] P. M. Fitts, "The information capacity of the human motor system in controlling the amplitude of movement," *Journal of Experimental Psychology*, vol. 47, no. 6, pp. 381–391, 1954.

[49] ISO, "Ergonomics of human-system interaction — Part 411: Evaluation methods for the design of physical input devices," Report Number ISO/TS 9241-411:2012, International Organisation for Standardisation, 2012.

[50] R. W. Soukoreff and I. S. MacKenzie, "Towards a standard for pointing device evaluation, perspectives on 27 years of fitts' law research in hci," *International Journal of Human-Computer Studies*, vol. 61, no. 6, pp. 751–789, 2004.

[51] H. Jiang, D. Weng, Z. Zhang, and F. Chen, "Hifinger: One-handed text entry technique for virtual environments based on touches between fingers," *Sensors*, vol. 19, no. 14, p. 3063, 2019.

[52] H. Jiang, D. Weng, X. Dongye, and Y. Liu, "Pinchtext: One-handed text entry technique combining pinch gestures and hand positions for head-mounted displays," *International Journal of Human–Computer Interaction*, vol. 0, no. 0, pp. 1–17, 2022.

[53] A. Gupta, C. Ji, H.-S. Yeo, A. Quigley, and D. Vogel, "Rotoswype: Word-gesture typing using a ring," in *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems - CHI '19*, (New York), pp. 1–12, ACM, 2019.

[54] A. Sluÿters, M. Ousmer, P. Roselli, and J. Vanderdonckt, "Quantumleap, a framework for engineering gestural user interfaces based on the leap motion controller," *Proceedings of the ACM SIGCHI Conference on Human-Computer Interaction*, vol. 6, pp. 1–47, 2022.

[55] F. M. Caputo and A. Giachetti, "Evaluation of basic object manipulation modes for low-cost immersive virtual reality," in *Proceedings of the 11th Biannual Conference on Italian SIGCHI Chapter – CHItaly '15*, (New York), pp. 74–77, ACM, 2015.

[56] I. Chatterjee, R. Xiao, and C. Harrison, "Gaze + gesture: Expressive, precise and targeted free-space interactions," in *Proceedings of the SIGCHI ACM on International Conference on Multimodal Interaction – ICMI '15*, (New York), pp. 131–138, ACM, 2015.

[57] I. A. S. Filho, E. N. Chen, J. M. da Silva Junior, and R. da Silva Barboza, "Gesture recognition using leap motion: A comparison between machine learning algorithms," in *Proceedings of SIGGRAPH '18 Posters*, pp. 1–2, New York: ACM, 2018.

[58] A. Clark and D. Moodley, "A system for a hand gesture-manipulated virtual reality environment," in *Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists – SAICSIT '16*, (New York), pp. 1–10, ACM, 2016.

[59] X. Jiang, Z. G. Xiao, and C. Menon, "Virtual grasps recognition using fusion of leap motion and force myography," *Virtual Reality*, vol. 22, pp. 297–308, 2018.

[60] M. Seixas, J. Cardoso, and M. T. G. Dias, "One hand or two hands? 2d selection tasks with the leap motion device," in *The Eighth International Conference on Advances in Computer–Human Interactions – ACHI '15*, (Lisbon, Portugal), pp. 33–38, IARIA, 2015.

[61] K. S. Jones, T. J. McIntyre, and D. J. Harris, "Leap motion and mouse-based target selection: Productivity, perceived comfort and fatigue, user preference, and perceived usability," *International Journal of Human–Computer Interaction*, vol. 36, no. 7, pp. 621–630, 2020.

[62] T. J. Dube, Y. Ren, H. Limerick, I. S. MacKenzie, and A. S. Arif, "Push, tap, dwell, and pinch: Evaluation of four mid-air selection methods augmented with ultrasonic haptic feedback," *Proceedings of the ACM SIGCHI Conference on Human-Computer Interaction*, vol. 6, no. ISS, pp. 207–225, 2022.

[63] M. C. B. Seixas, J. C. Cardoso, and M. T. G. Dias, "The leap motion movement for 2d pointing tasks: Characterisation and comparison to other devices," in *International Conference on Pervasive and Embedded Computing and Communication Systems – PECCS '15*, (New York), pp. 15–24, IEEE, 2015.

[64] P. Rittitum, W. Vatanawood, and A. Thongtak, "Digital scrum board using leap motion," in *IEEE/ACIS 15th International Conference on Computer and Information Science – ICIS*, (New York), pp. 1–4, IEEE, 2016.

[65] D. Bachmann, F. Weichert, and G. Rinkenauer, "Evaluation of the leap motion controller as a new contact-free pointing device," *Sensors*, vol. 15, no. 1, pp. 214–233, 2014.

[66] A. Zocco, M. D. Zocco, A. Greco, S. Livatino, and L. T. De Paolis, "Touchless interaction for command and control in military operations," in *Proceedings of Augmented and Virtual Reality: Second International Conference*, (New York), pp. 432–445, Springer, 2015.

[67] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," in *Advances in Psychology*, vol. 52, pp. 139–183, Amsterdam: Elsevier, 1988.

[68] S. Kauppinen, S. Luojus, J. Tuomisto, and A. Ahlgren, "Utilizing gesture recognition technology in children's interactive storybook," in *Proceedings of International Conference on Making Sense of Converging Media – AcademicMindTrek '13*, (New York), pp. 76–79, ACM, 2013.

[69] M. Renzi, S. Vassos, T. Catarci, and S. Kimani, "Touching notes: A gesture-based game for teaching music to children," in *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction – TEI '15*, (New York), pp. 603–606, ACM, 2015.

[70] H. Liang, J. Chang, I. K. Kazmi, J. J. Zhang, and P. Jiao, "Hand gesture-based interactive puppetry system to assist storytelling for children," *The Visual Computer*, vol. 33, pp. 517–531, 2017.

[71] T. Alzubi, R. Fernández, J. Flores, M. Duran, and J. M. Cotos, "Improving the working memory during early childhood education through the use of an interactive gesture game-based learning approach," *IEEE Access*, vol. 6, pp. 53998–54009, 2018.

[72] L. Zaina, E. Castro, S. Martinelli, and T. Sakata, "Educational games and the new forms of interactions: Exploring the use of hand gestures in a computational thinking game," *Smart Learning Environments*, vol. 6, pp. 1–17, 2019.

[73] M. M. Bradley and P. J. Lang, "Measuring emotion: The self-assessment manikin and the semantic differential," *Journal of Behavior Therapy and Experimental Psychiatry*, vol. 25, no. 1, pp. 49–59, 1994.

[74] M. S. A. Rahman, N. M. Ali, and M. Mohd, "A study on the naturalness of gesture-based interaction for children," in *Proceedings of the Third International Visual Informatics Conference - IVIC '13 (LNCS 8237)*, (Cham), pp. 718–728, Springer, 2013.

[75] S. Fallah and S. Mackenzie, "H4vr: One-handed gesture-based text entry in virtual reality using a four-key keyboard," in *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems – CHI EA '23*, (New York), pp. 1–7, ACM, 2023.

[76] I. S. MacKenzie and R. W. Soukoreff, "Phrase sets for evaluating text entry techniques," in *Extended Abstracts of the ACM SIGCHI Conference on Human Factors in Computing Systems - CHI '03*, (New York), p. 754–755, ACM, 2003.

Appendices

The questionnaire used in the first user study

1. Gender: _____
2. Age: _____
3. Is English your first language? _____
4. Have you had any experience typing in VR? _____
5. Have you had any experience using this keyboard? _____
6. How would you rank your hand fatigue throughout this experiment from 1 to 7?
   Flat: 1 (no fatigue) - 7 (very fatiguing) _____
   Cross: 1 (no fatigue) - 7 (very fatiguing) _____
7. Which keyboard layout would you prefer to use?
   Flat
   Cross
   Same _____
8. Which keyboard do you think had better performance?
   Flat
   Cross
   Same _____
9. Which keyboard do you think had a lower error rate?
   Flat
   Cross
   Same _____
10. Which keyboard required more visual search?
    Flat
    Cross
    Same _____
11. Do you think your hand fatigue was reduced through the sessions?
    Yes
    No _____

12. Do you think your speed rate improved through the sessions?
    Yes
    No _____

13. Do you think your error rate was reduced through the sessions?
    Yes
    No _____

The questionnaire used in the second user study

Participant Information:

1. Participant ID: _____
2. Gender: _____
3. Age: _____

LMC:

1. On a scale of 1 to 10, how comfortable were you using the LMC to select targets (1 = very uncomfortable, 10 = very comfortable)? Rating: _____
2. What did you like about the LMC for selecting targets?

3. What did you find challenging or uncomfortable about the LMC for selecting targets?

4. Were there any specific situations or tasks where the LMC excelled or fell short? Please describe.

5. Do you have any suggestions for improving the LMC?

Leapboard:

1. On a scale of 1 to 10, how comfortable were you using the Leapboard to select targets (1 = very uncomfortable, 10 = very comfortable)? Rating: _____
2. What did you like about the Leapboard for selecting targets?

3. What did you find challenging or uncomfortable about the Leapboard for selecting targets?

4. Were there any specific situations or tasks where the Leapboard excelled or fell short? Please describe.

5. Do you have any suggestions for improving the Leapboard?

Touchpad:

1. On a scale of 1 to 10, how comfortable were you using the touchpad to select targets (1 = very uncomfortable, 10 = very comfortable)? Rating: _____
2. What did you like about the touchpad for selecting targets?
3. What did you find challenging or uncomfortable about the touchpad for selecting targets?
4. Were there any specific situations or tasks where the touchpad excelled or fell short? Please describe.
5. Do you have any suggestions for improving the touchpad?

Overall Comparison:

Rank the three methods (LMC, Leapboard, touchpad) from your most preferred to least preferred for selecting targets. Most Preferred:

_____ Middle Preference: _____ Least Preferred: _____

General Feedback:

Do you have any additional comments or observations about your overall experience using these input methods in the study?

Thank you for participating in this user study! Your feedback is valuable and will help improve user interactions with these input methods.