Over the Mouse: Navigating across the GUI with Finger-Lifting Operation Mouse

YoungIn Kim
HCI Lab
School of Computing, KAIST
Daejeon, Republic of Korea
youngin@kaist.ac.kr

Yohan Yun
HCI Lab
KAIST
Daejeon, Republic of Korea
yohanme@kaist.ac.kr

Taejun Kim
HCI Lab
School of Computing, KAIST
Daejeon, Republic of Korea
taejun.kim@kaist.ac.kr

Geehyuk Lee
HCI Lab
School of Computing, KAIST
Daejeon, Republic of Korea
geehyuk@gmail.com

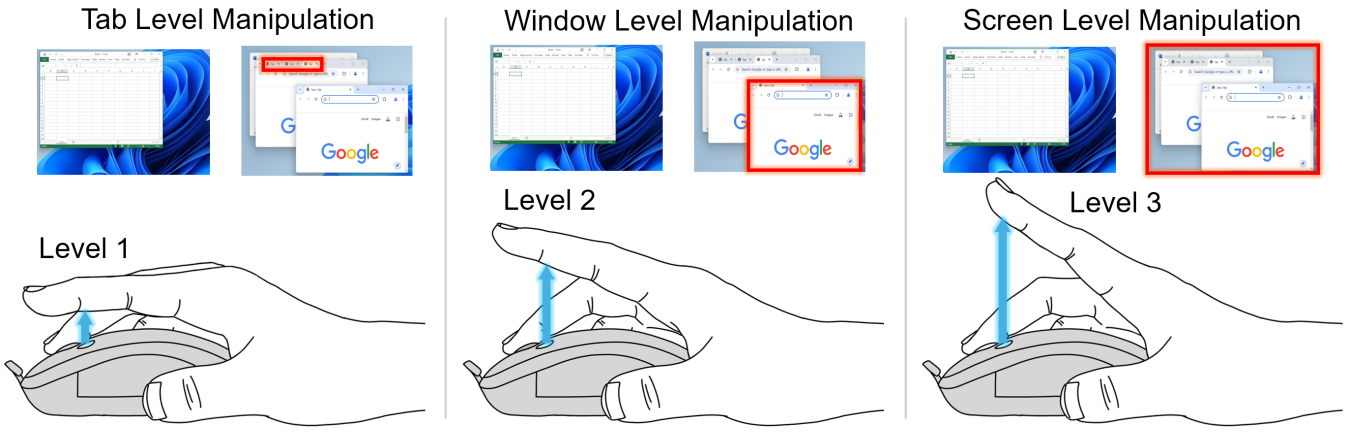

Figure 1: 'Over the Mouse' enables users to navigate across different levels of the GUI hierarchy. The illustration shows a user accessing various levels of a window hierarchy, including tabs, windows, and screens, by adjusting the height of the finger over the mouse.

Abstract

Modern GUIs often have a hierarchical structure, i.e., the z-axis of the GUI interaction space. However, conventional mice do not support effective navigation along the z-axis, leading to increased physical movements and cognitive load. To address this inefficiency, we present the OtMouse, a novel mouse that supports finger-lifting operations by detecting finger height through proximity sensors embedded beneath the mouse buttons, and 'Over the Mouse' (OtM) interface, a set of interaction techniques along the z-axis of the GUI interaction space with the OtMouse. Initially, We evaluated the performance of finger-lifting operations (n = 8) with the OtMouse for two- and three-level lifting discrimination tasks. Subsequently, we conducted a user study (n = 16) to compare the usability of the OtM interface and traditional mouse interface for three representative tasks: 'Context Switch,' 'Video Preview,' and 'Map Zooming.' The results showed that OtM interface was both qualitatively and quantitatively superior to using traditional mouse in the Context Switch and Video Preview tasks. This research contributes to the ongoing efforts to enhance mouse-based GUI navigation experiences.

CCS Concepts

• **Hardware** → *Communication hardware, interfaces and storage.*

Keywords

Input device, Augmented mouse, Finger-lifting operations

CHI '25, Yokohama, Japan

ACM ISBN 979-8-4007-1394-1/25/04
https://doi.org/10.1145/3706598.3713340

ACM Reference Format:
YoungIn Kim, Yohan Yun, Taejun Kim, and Geehyuk Lee. 2025. Over the Mouse: Navigating across the GUI with Finger-Lifting Operation Mouse. In *CHI Conference on Human Factors in Computing Systems (CHI '25), April 26–May 01, 2025, Yokohama, Japan.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3706598.3713340

1 Introduction

Modern Graphic User Interfaces (GUIs) often feature a hierarchical structure with multiple interface levels. For instance, a screen may have several open applications, such as Chrome or Excel, each has multiple tabs, and each tab contains various UI layout elements, such as list panes and document panes. This forms a hierarchy that spans from the screen (top level) to the individual panes (bottom level). Similarly, a menu often groups related commands into submenus, creating another form of hierarchy. These hierarchies can be conceptually denoted as the "z-axis" arrangement of the conventional 2D GUI elements. Hierarchical structures are a hallmark of modern GUIs, and supporting efficient navigation within these hierarchies is essential to ensure their usability.

However, the current mouse interfaces have limitations in efficiently navigating GUI hierarchies. For example, when using a web browser like Google Chrome, users should frequently move the cursor between the central content area and the tab or taskbar areas at the top or bottom of the screen to switch between application windows or document tabs. These frequent cursor movements can increase task time and physical strain. One possible solution may be adding additional buttons to the thumb side of the mouse [28, 29]. While this approach may work for a few command shortcuts, it may not scale properly as the number of supported commands increases. Increasing the number of buttons is challenging due to the limited surface area within easy reach of the thumb on a mouse, and also burden users by requiring them to memorize more button-command mappings.

We sought an answer to the current challenge in other previous works on augmenting the input space of the mouse, which has a long history. Two notable cases were multi-touch sensing mice [49, 50] and force-sensing mice [12, 16, 31, 47, 53]. Multi-touch sensing mice support multi-touch gestures to facilitate switching between applications and between tabs. However, since their gestures are performed on horizontal plane, we anticipate that they may not support intuitive z-dimensional navigation for GUI hierarchies. Meanwhile, some of the force-sensing mice [16, 53] support vertical operations, normal to the mouse plane, and may support intuitive z-dimensional navigation. Their continuous force-input space may also correspond to the expandable levels in GUI hierarchies. However, their force-input space physically conflict with the existing mouse button input space, harming existing functionality. While other force-sensing mouse studies proposed positioning a force sensor on the thumb side [12] or using an inflatable form [31], their force operations were not made in vertical, z-axis directions.

The potential of vertical operations in force-sensing mice inspired us to consider vertical interactions in the input space, but in the opposite direction of "pushing" force operations. The result was *OtMouse*, which can detect finger-lifting operations using Time-of-Flight (ToF) sensors mounted beneath the left and right mouse buttons. With the OtMouse, users can perform finger-lifting operations to access different levels within GUI hierarchies while simultaneously moving the mouse. These lifting operations can function as any metaphorical z-dimensional input, such as selecting a graphical object in a z-order arrangement or adjusting the font size of selected text.

To optimize the potential of the new input space enabled by the OtMouse, we conducted a preliminary experiment to determine users' preferred lift heights and the number of lift levels. We then carried out a user study to assess the speed and accuracy of lift operations. Building on these user data, we developed a series of interaction techniques, collectively referred to as the 'Over the Mouse' (OtM) interface, enabled by the *OtMouse*. Finally, we evaluated the OtM interface using representative example task scenarios. Results, including SUS scores and user interviews, indicated that the OtM interface provided superior usability compared to a traditional mouse interface in most cases.

The contributions of this study are two-fold:

- The *OtMouse*, a novel mouse supporting finger-lifting operations.
- Empirical validation of the benefit of the OtM interface enabled by the *OtMouse*.

Section 2 reviews related studies on augmented mice and the z-dimensional characteristics of the desktop GUIs. Section 3 describes the design and implementation of the *OtMouse* in detail. Section 4 presents the evaluation of the finger-lifting operations. Section 5 introduces the OtM interaction scenarios, and Section 6 presents the evaluation of these techniques through selected scenarios in comparison with a conventional mouse interface.

2 Background

OtMouse is an augmented mouse that extends the dimension of traditional xy-plane input by using finger-lifting gestures. Therefore, in this section, we detail the augmented mice with z-axis interface studied on various platforms. In the end, we discuss the z-axis characteristics of the PC GUI covered in our user study.

2.1 Z-Axis Interface

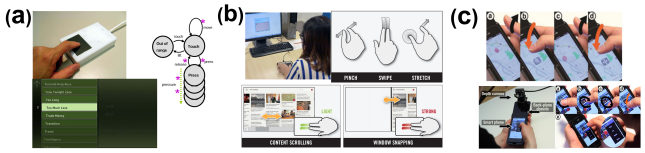

Figure 2: Previously proposed z-axis interaction using a pressure-/hover-based interface. (a) glimpse state suggested by Forlines et al. (2005) [16], (b) Pressture interaction scenarios proposed by Rendl et al. (2014) [45], (c) Air+Touch gesture and representative scenarios (map, menu) proposed by Chen et al. (2014) [14]

The z-axis interface enables sequential engagement and enhances spatial cognition of the physical environment, facilitating rapid and intuitive control over an additional input dimension beyond the traditional xy-plane [30]. Here, spatial cognition refers to the cognitive representation of positional relationships in the surrounding space [21], and sequential engagement refers to the interaction of maintaining the sense of direction in a changing environment [35]. Representative examples of z-axis interface implementations include the pressure-based and hover-based interfaces.

2.1.1 Pressure-based interfaces. Early forms of pressure input were single-input devices in the shape of a stylus [4, 34, 41, 42]. Ramos et al. experimented with continuous pressure sensing values as discrete multi-level input on pen-based platforms [41]. Blasko and Feiner implemented multi-point pressure detection across four touchpad areas on laptops, demonstrating control of 14 widgets without visual feedback [4]. These works show the potential of multi-level z-axis input for GUI control. Rekimoto and Schwesig proposed 'Presensell', a system that recognizes contact area and force on a 2D touchpad [42]. They suggested use cases for map zooming and menu control using pressure input. 'Presstures' (Figure 2(b)) extended existing touchpad gestures with pressure, measured accuracy for two-level and three-level pressure inputs, and presented use cases for window snapping [45].

There are also studies that explored the application of pressure input in mice. Zeleznik et al. introduced a 'pop-through' state on the mouse click buttons to create an expandable menu, a form of zooming interface [53]. Forlines et al. (Figure 2(a)) proposed the 'glimpse state,' a state machine for pressure-based multi-level input that facilitates quick transitions between content and tool pane hierarchy levels [16]. Cechanowicz et al. designed a mouse that detects pressure input outside the click area to resolve interference issues with traditional click inputs in earlier pressure-sensitive mice [12]. These studies suggest that integrating a z-axis interface into a mouse can be mapped in various ways to utilize z-axis GUI.

2.1.2 Hover-based interfaces. User interfaces utilizing hover gestures have been researched for various environments, such as tabletop and mobile [2, 23, 25, 46, 48]. Hilliges et al. used IR cameras in tabletop environments to detect hand gestures and depth, allowing interactions like virtual object manipulation and bi-manual stretching. [25]. Takeoka et al. proposed a similar device and utilized it for multi-layer drawing tools and zooming scenarios in maps [48]. Chen et al. (Figure 2(c)) explored the use of hover gestures on smartphones using a depth camera to recognize gestures for map zooming control, scroll speed adjustment, and menu expansion [14]. Heo et al. developed a touchpad that recognizes pressure and hover and demonstrated it on video browsing and object manipulation scenarios, comparing performance, time, and user preferences with the button and traditional touchpad methods [23]. Consequently, these studies show that previous z-axis interfaces have introduced z-axis GUI control through a multi-level input method. Accordingly, our study's evaluation will also focus on z-axis GUI control based on lifting levels among the various operation methods that the OtM interface can provide.

For desktop hover gestures, Franz et al. integrated a Leap Motion device onto a mouse to enable hover gestures above the mouse [17]. However, since using these gestures required users to remove their hand from the mouse, this approach was more akin to facilitating rapid switching between two input devices rather than expanding the input space of a single device. Beyond these examples, some studies have explored expanding a mouse's input capabilities using methods other than pressure or hover input.

2.2 Augmented Mice

The simplest way to additional input is by adding physical buttons. Various manufacturers have released mice with multi-buttons beyond the standard buttons [28, 29]. These buttons are widely used for shortcuts or specific macros. On the other hand, several studies introduced in Section 2.1.1 used pressure input as a z-axis interface to expand the 2D input of mice with related scenarios. For examples other than pressure input, Rockin'Mouse featured a curved base and an internal tilt sensor, enabling efficient 3D manipulation by tilting the mouse [3]. Roly-Poly Mouse had a hemispherical base and assisted with 3D manipulation by allowing the mouse to be rotated or tilted [38]. VideoMouse incorporated a camera on its underside for 6-DoF sensing, enabling functions such as object manipulation, zooming, and layer-based interfaces [26].

However, these methods have limitations for precise and fast input. Traditional multi-button mice require users to memorize the command associated with each button based solely on its spatial position—without any semantic connection to its corresponding function—and to reposition their hand in order to select a function. [12]. Additionally, the thumb position may shift as the users manipulate the mouse, leading to unintended or incorrect button inputs. Devices like Rockin'Mouse, Roly-Poly Mouse, and VideoMouse may interfere with standard interactions due to tilting or lifting actions, and their reliance on large gestures makes them unsuitable for frequent use. Pressure-based input also faces several challenges. Zeleznik's research highlighted how constant clicking interferes with multi-level pressure input, as the pressure required for clicks disrupts the default click action [53]. Cechanowicz's device has similar challenges to multi-button mice, where keeping buttons within thumb reach reduces thumb repositioning but increases the risk of unintended inputs.

In contrast, finger-hovering gestures on a mouse may address these issues. The click location remains consistently accessible, and lifting the finger is clearly distinctive from clicking, therefore preserving existing functions and preventing incorrect inputs caused by mouse movement. Additionally, this approach leverages the intuitiveness of finger gestures for z-axis level selection. Based on this concept, we devised the OtM interface utilizing finger lift gestures.

2.3 Z-Axis in GUI

Desktop GUI often features either a structural hierarchy (Figure 3(a)), or a hierarchy of the displayed information (Figure 3 (d)). Additionally, software like *Photoshop* or *Premiere* incorporates vertical layers, enabling users to independently edit specific details within broader content (Figure 3 (b)). The verticality of these hierarchies and layers can be conceptualized as a z-axis—an orthogonal axis relative to the existing xy-plane [52]. This section discusses UIs with z-axis structures and examines the limitations of existing interaction methods for engaging with such UIs.

2.3.1 Window hierarchy. In a desktop environment, users have a hierarchical workspace called a virtual screen. As shown in Figure 3-(a), a virtual screen hosts multiple windows, each with panes containing various content. Users use this window hierarchy to conveniently navigate through similar functions placed within the same level. For example, users can consecutively click on browser tabs placed sequentially in the same pane within a browser window.

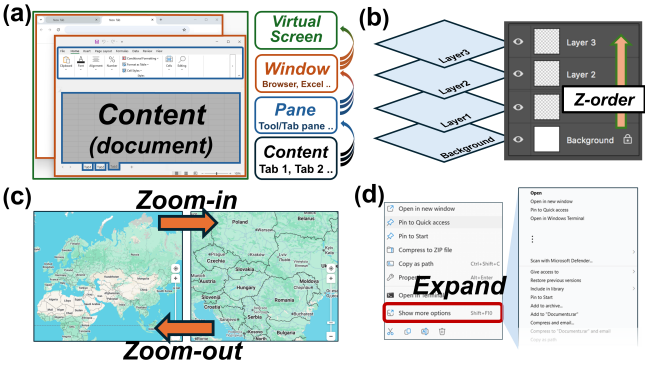

Figure 3: Examples of GUIs with z-axis characteristics. (a) Window hierarchy, (b) Layer-based interface, (c) Graphical zooming interface, (d) Semantic zooming interface.

However, the current desktop input system may not fully support interaction between different window hierarchy levels. For instance, level switching between documents and tabs requires repetitive cursor movement between the document content and the tab-pane. Such cursor movements often result in physical burden and cognitive load [5, 8, 32]. Therefore, a z-axis interface system that can enable direct hierarchical selection can be powerful for desktop environments.

2.3.2 Layer-based interfaces. Layer panels are commonly used in UI design tools or media editing software. These layers facilitate non-destructive editing[1], allowing users to modify individual elements independently while maintaining a visual representation of the complete result. The stacking of higher layers above lower ones, which visually occludes the content beneath, aligns with the concept of a z-axis in the z-order metaphor.

Despite this, layer interfaces often require excessive mouse movements to reorder or select specific layers. To address this inefficiency, research is needed to develop systems that intuitively leverage the z-axis characteristics of layers. Such systems could also benefit 3D object selection. Many studies have explored 3D selection using a mouse [36, 40, 51, 54]. By representing physical depth with layers, an input system capable of directly manipulating the z-axis could offer a novel approach to 3D selection.

2.3.3 Zooming interfaces. Zooming interfaces allow users to control the level of information displayed, either graphically or semantically, as shown in Figure 3-(c), (d). For instance, word processors and web browsers enable users to zoom in on the physical size of images or text, while file explorers support semantic zooming by expanding menus. Video players allow users to zoom across different control scales, such as adjusting the amount of time to skip forward. However, current input methods limit zoom control to either mouse wheels or a series of mouse clicks. Since scroll wheels typically manage y-axis movement in UIs, some applications require hotkeys for zooming, and others may even need multiple hotkeys for multi-level scale adjustments (e.g., video players) . To address these limitations, the z-axis interface could offer a more intuitive solution for zoom control.

Another example of zooming interfaces is adjusting the control-display (CD) gain of a mouse, which is the ratio of cursor movement on the screen to the physical movement of the mouse. Low CD gain on large displays necessitates frequent clutching to reach distant targets, while high CD gain compromises precision for small object selection [37]. To address this, previous research has explored dynamic CD gain adjustments based on context or mouse speed [11, 15, 18]. Therefore, mapping CD gain to the z-axis can facilitate dynamic and intuitive CD gain adjustment.

3 Designing OtMouse

OtM interface aims to utilize finger-lifting action with various GUI interactions. This section explains the implementation of OtMouse and the parameters defining the finger-lifting actions on the *OtMouse.*.

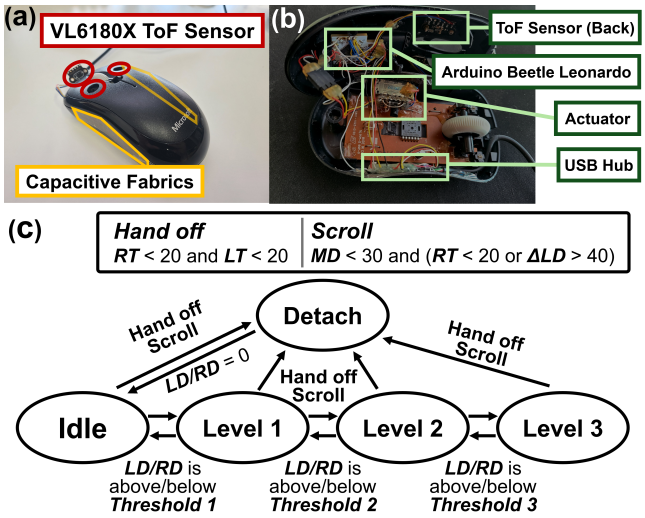

Figure 4: (a) The appearance of OtMouse. (b) The internal structure and components of *OtMouse*. (c) The state machine of *OtMouse*. 'RT' and 'LT' denote the right/left touch sensor values, while 'LD', 'MD', and 'RD' denote the left/middle(wheel)/right tof sensor values. Values are derived from the data collection of Section 3.3.

3.1 Mouse for Finger Lift

The z-axis finger-lifting on a mouse means raising the index or middle finger from the mouse buttons to a specific height. Finger height can be inputted as a continuous value or a discrete value with defined transition thresholds. Continuous values are suitable for directly controlling GUI elements (e.g., volume or sliders), while discrete values are better suited for shortcuts or multi-level z-axis GUI interactions. In this study, discrete level selection was employed to examine multi-level z-axis GUI interactions. Key parameters included finger configurations, the number of z-axis levels, and height thresholds for transitions between levels. We chose three finger configurations: index finger (left), middle finger (right), and both fingers simultaneously. For dual-finger lifts, the height was determined by the higher of the two fingers to ensure clear differentiation of lift

actions. For z-axis level selection, two or three levels were chosen. Excessive levels would require significant finger-lifting and precise height control, increasing physical and cognitive strain [20, 30, 33]. The lift heights for each level were determined experimentally, as described in Section 3.4.

Our initial approach for detecting finger height was to use a small camera. However, this required positioning the camera at the back of the mouse to capture fingers, limiting the mouse design to a flat shape. Instead, we decided to drill holes in the mouse's click areas and place Time-of-Flight (ToF) sensors underneath to detect the height. This method is simple and works with standard mouse designs. However, simply detecting finger height may trigger unintended level selection. To prevent this, we added touch sensors to the sides and additional proximity sensors to the front for false positive lift detection, as detailed in Section 3.3.

3.2 Implementation

OtMouse is based on the Microsoft Basic Optical v2.0 mouse with three VL6180X ToF sensors. VL6180X uses a tiny laser to measure the distance between the sensor and the surface spot in the vertical direction, offering 1mm resolution over a range of 0 to 100mm. Three sensors are mounted to face upward from both click areas and toward the scroll wheel from the front. Each value is smoothed using a 1euro filter [10], and the Beetle Leonardo board inside the mouse handles this processing by reading the sensor data.

The capacitive fabrics were attached to both sides of the *OtMouse*, which functions as a touch sensor, detecting capacitance values via the ADC port of the Leonardo board MCU. Power for the MCU and data communication with the PC is handled through a USB hub integrated into the mouse, utilizing original cable. The state machine for each finger configuration is implemented in the Leonardo board, classifying states based on distance and touch data, as shown in Figure 4 (c). This state machine operates at a frequency of 20 Hz. The data collection process and the state transition criteria are detailed in Section 3.3. When a state transition occurs, a Nintendo Switch Actuator connected to the TX pin provides haptic feedback through a 0.7ms 5V pulse input. For both lift actions, level 1 is activated when both fingers are lifted to the threshold simultaneously, and the state proceeds on the position of the higher finger.

3.3 Refining State Machine via Sensor Data Collection

Data collection in everyday usage scenarios was firstly conducted to determine transition conditions for the state machine. As noted in Section 3.1, relying solely on finger height can result in unintended level selections, such as when the hand is released from the mouse or the scroll wheel is used, negatively impacting usability. Therefore, setting appropriate transition conditions using additional sensors is essential to prevent these malfunctions.

We collected mouse usage data from six participants (2 females, 4 males; age: M = 24.5 years, SD = 2.7). All participants were right-handed and used the mouse in a standard desktop setup with a 27-inch monitor, keyboard, and mouse. The data collection was performed on three tasks:

- Creating a *PowerPoint* slide: Participants created a slide like the provided example designed to encourage the use of functions such as shape, font, color, and border adjustment.
- Shopping: Participants searched for a recipe for a given food on *Google*, found the ingredients on e-commerce.
- Game playing: Participants played *Angry Birds* for 5 minutes.

These tasks were selected to observe 'Clicking at various points,' 'Changing hand position when using scroll wheel', 'Resting the hand away from the mouse to use the keyboard,' and 'Cursor dragging.' Each task lasted up to five minutes. During the tasks, we recorded the following data at 20Hz: 'clicks of each mouse button (left, right, wheel),' 'mouse wheel scroll and direction,' 'cursor position,' 'ToF sensor values,' and 'touch sensor values'

Based on the data collected from task performance and threshold preferences outlined in Section 3.4, we defined the states of OtMouse as **Detach**, **Idle**, and **Lifted**. **Detach** represents inadvertent finger-liftings exceeding the height threshold during basic mouse operations, while **Lifted** indicates deliberate finger-lifting to a specific level. Then, we identified that **Detach** situations occur when users either release their hand from the mouse or operate the scroll wheel. Based on this, we established criteria to filter these two scenarios. Our observations revealed that when users scroll with their middle finger, the palm slightly lifts from the right touch sensor, registering approximately 10% (value: 30) of the **Idle** sensor value (value: 300). Conversely, during scrolling with the index finger, the left ToF sensor delta exceeded 40, while the front ToF sensor distance dropped below 30 mm. When user releases hand from the mouse, the values of both touch sensors dropped below 20. These values were used as criteria for transitions between states—**Idle** to **Lifted**, or any level to **Detach**—as depicted in Figure 4(c).

3.4 Understanding User Preference for Finger-Lifting Heights

We conducted this experiment to determine the optimal lift height for the *OtMouse*, where users can easily distinguish without discomfort. We gathered the preferred height response for each lift action from 8 participants (2 females, 6 males; age: M = 24.6 years, SD = 2.2). All participants were right-handed mouse users. The height was calculated by subtracting the distance between the ToF sensor and the mouse surface from the sensor value.

In the experiment, participants used keyboard number keys to select the lift level to adjust and arrow keys to modify the threshold height for state transitions. Using their left hand, they first selected the level and then adjusted the threshold height of level in 1 mm increments within a 5 to 100 mm range using the up/down arrow keys. They were asked to select a 'comfortable' and 'easily distinguishable' point for each level. They tested the adjusted thresholds by lifting their fingers on the mouse and repeated the process until satisfied. This procedure was conducted for both two-level and three-level discrimination across three finger configurations (index, middle, and both), resulting in six combinations. Each combination was repeated over two cycles. The entire experiment took approximately 30 minutes.

Figure 5 shows the mean and range of preferred lift heights reported by participants. Notably, the maximum height for a certain level never exceeded the minimum height of the subsequent level.

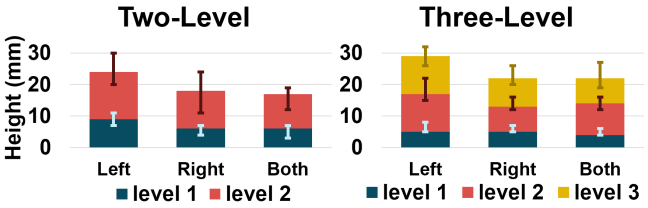

Figure 5: The preferred height values responded by users for the two-level(left) and three-level(right) discrimination. The values in the graph represent the average transition threshold for each level (cumulative), and error bars indicate the response range for each level.

In all cases, lift height responses remained below 33 mm, approximately 30% of the ToF sensor's detectable range. Preferred heights were highest for the index finger, while the middle and both fingers had similar value, likely due to the relative ease of lifting the index finger. These findings enabled the determination of lifting level threshold in the state machine (Figure 4(c)).

4 User Study 1: Analyzing Speed and Accuracy in Performing Lift Gestures

This experiment was conducted to measure the completion time (CT) and error rates (ER) in 2- and 3-level finger lift combined with selection gestures. Four directional stokres of mouse movements were performed subsequent to finger-lifting to finalize a task trial.

4.1 Level Selection Gestures

We used up, down, left, and right strokes on the mousing plane to select a desired level after lifting a finger from the *OtMouse*. These directions are the most intuitive for users, leveraging spatial cognition to reduce task complexity. [30]. In this gesture, up refers to stroking the mouse toward the monitor, while down refers to stroking the mouse toward the body. Additionally, stroke gestures can be used for pie-menu navigation or be intuitively mapped to horizontal interactions with the GUI.

4.2 Task

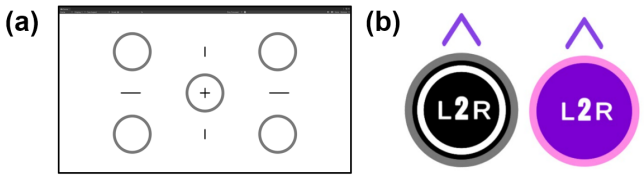

Figure 6: (a) The five locations where the target appears. (b) The appearance of the target before activation (left) and after activation (right). The target represents a gesture where both fingers are lifted to level 2, followed by an upward stroke.

Eight participants (1 female, 7 males; Age: M = 26.1 years, SD = 3.2), all right-handed mouse users, were recruited for the experiment. Before beginning, participants tested the lift heights established in Section 3.4 and confirmed they were sufficiently distinguishable without adjustments. The experiment was conducted under a desktop setup with a 27-inch monitor, keyboard, and *OtMouse*. Participants were tasked with moving the cursor to a target and performing the indicated gesture. Targets were 8 cm diameter circles that appeared randomly in one of five locations: the screen center or the centers of the four quadrants based on the screen's center (Figure 6(a)). This random positioning tested the stability of gestures during continuous mouse movement. Each target's inner region specified the finger configuration (left, right, or both) and lift level (1–2 for two-level and 1–3 for three-level discrimination), while the outer region indicated the stroke direction (Figure (b)). A stroke was recognized when the cursor moved outside the target boundary. Trials were marked as correct only if the finger configuration, lift level, and stroke direction were accurate; otherwise, they were recorded as errors. Errors incurred a two-second penalty time and auditory feedback. Participants were given a one-second buffer period to perceive the target's information, during which the target appeared in monotone (Figure 6(b), left). If the cursor remained on the target during buffer time, the target's appearance changed to the one shown on the right of Figure 6(b). CT was measured from this point to the end of the stroke.

Participants completed four blocks per discrimination level, with at least 60 seconds of rest between blocks. The order of discrimination level was counterbalanced, and the experiment lasted approximately 60 minutes per participant.

4.3 Results and Discussion

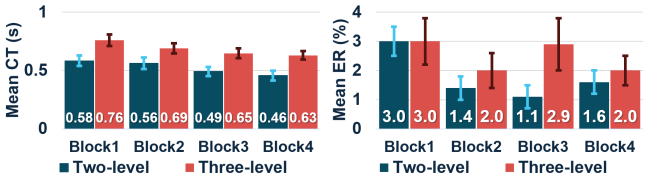

Figure 7: The result of CT and ER for gestures according to blocks. Error bars show standard error.

We collected a total of 5,760 trials ((2+3) lift levels × 3 finger configurations × 4 directions × 4 blocks × 3 repetitions × 8 participants). A two-way repeated measures ANOVA (RM-ANOVA) was conducted on CT and ER to examine the learning effect based on *Block* and *Discrimination level*. For CT, significant differences were found for both *Discrimination level* (F (1, 7) = 98.379, p < .001) and *Block* (F (3, 21) = 12.390, p < .001), without significant interaction effect. A post-hoc test using a paired-samples T-Test for Block revealed significant differences between Block1-Block4 (t(7) = 5.553, p < .001, with bonferroni correction) and Block2-Block4 (t(7) = 3.579, p = .011) pairs. For ER, no significance was found for either Discrimination level or Block, with no interaction effect. Due to the significant difference in CT over Block, subsequent analyses are conducted for data averaging Block3 and Block4. A paired-samples T-Test on the average CT and ER revealed a significant difference

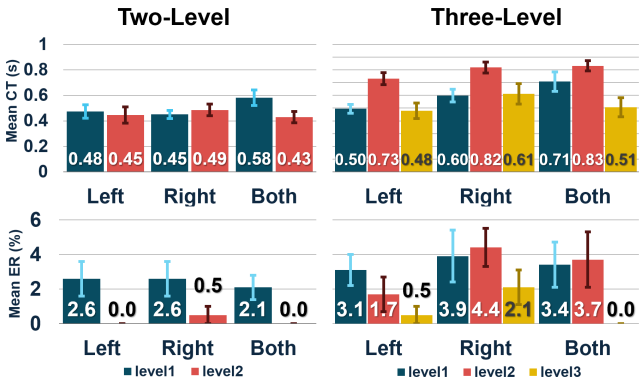

Figure 8: The results of CT and ER for gestures according to lift finger and level based on two types of discrimination levels. Each value is an aggregate of the average of the values from Blocks 3 and 4. Error bars show standard error.

between two-level (0.475s) and three-level (0.638s) discrimination ($t(7)$ = -8.364, p < .001). For ER, the difference between two-level (1.3%) and three-level (2.4%) was not statistically significant.

We conducted a two-way RM-ANOVA on CT to examine differences in Lift Finger and Lift Level for each discrimination level (Figure 8). ER was analyzed separately due to zero values in some categories. Both CT and ER were analyzed by averaging the values from Block 3 and 4. In two-level discrimination, RM-ANOVA on CT showed no significant differences for Lift Finger or Lift Level, but a significant interaction effect was found (F (2, 14) = 7.996, p = .005). A post-hoc analysis using the paired-samples T-Test revealed a statistically significant difference between 'level 1 both lift' and 'level 2 both lift' ($t(14)$ = 3.474, p = .048, with bonferroni correction). For ER, due to normality violations, the Wilcoxon signed-rank test was used for Lift Level, showing a significant difference (W = 43.5, p = .015), while the Friedman test for Lift Finger found no significance ($\chi^2(2)$ = 0.867, p = .648). We found that overall accuracy was higher with level 2 lift, likely due to the absence of upper limits in level 2.

In three-level discrimination, RM-ANOVA on CT showed statistical significance for both Lift Finger (F (2, 14) = 5.383, p = .018) and Lift Level (F (2, 14) = 14.394, p < .001), without significant interaction effect. Post-hoc analysis using the paired-samples T-Test found significant differences between left-right ($t(14)$ = -2.767, p = .045) and left-both ($t(14)$ = -2.911, p = .034) for lift finger, and between level 1 and level 2 ($t(14)$ = -3.829, p = .006) as well as level 2 and level 3 ($t(14)$ = 5.169, p < .001) for Lift Level. For ER, a one-way RM-ANOVA for Lift Finger showed no significant results, but the Friedman Test for Lift Level revealed significance ($\chi^2(2)$ = 9.176, p = .01). Wilcoxon signed-rank test showed significant differences between level 1 and level 3 (W = 28.0, p = .022).

In three-level discrimination, both finger and lift height influenced CT. Lifting with the index finger was faster, and selecting level 2 proved more challenging than other levels. This indicates that implementing more than four lift gesture levels may disproportionately increase CT for middle levels compared to the bottom or top levels. Additionally, the ER for level 1 was significantly higher than for level 3 though they had similar CT, suggesting that gesture speed does not always correlate with accuracy. This discrepancy may stem from the lack of an upper limit for level 3.

Overall, we demonstrated that combining finger-liftings with mouse stroke actions allows for 24 gestures to be performed within an average of 0.5 seconds with a 1.3% error rate and 36 gestures within 0.7 seconds with a 2.4% error rate. Given that every gesture accuracy exceeding 95%, we utilized both two- and three-level discrimination mechanism into the OtM interface. These stroke gestures were utilized as a pie-menu for z-axis GUI interactions.

5 OtM Interface: Interaction Techniques with OtMouse

The OtM interface aims to control the GUI by raising specific fingers to certain heights, and the height of the fingers can be recognized as either continuous or discrete values. This section introduces scenarios where *OtMouse* is used to control the GUI with continuous or discrete value inputs. Especially in discrete interaction, the scenarios are mainly focused on z-axis GUI interaction, and these are built on the gestures from User Study 1. Since two-level discrimination showed better CT and ER than three-level, two-level gestures are prioritized unless three-level z-axis interaction is necessary. The specific operation process of the proposed scenario can be seen through our accompanying video.

5.1 Finger-Lifting as Continuous Input for GUI Control

Continuous finger height can be used to manipulate continuous values in GUI widgets, where the height naturally maps to the value being controlled. However, since the minimum distinguishable height provided by *OtMouse* is 1 mm, noise may occur due to subtle finger tremors. Therefore, it is necessary to consider gain adjustments for continuous input.

GUI slider control. In PC environments, sliders are commonly found in GUIs for adjusting brightness, volume, or settings in software like *Photoshop*. Typically, these sliders require clicking and dragging the handle horizontally with a mouse to adjust the values. However, with the OtM interface, users can control the values by simply adjusting the height of their fingers over the slider, eliminating the need to click the handle. For slider control, height can be applied using either absolute control, assigning values directly from finger height, or relative control, modifying values based on finger displacement.

Continuous parameter control. Documentation programs, such as *PPT* and *Word*, allow users to adjust parameters like font size and line width. By utilizing the OtM interface, these parameters can be continuously scaled up or down without direct clicks, providing a seamless adjustment experience. This interaction can follow the typical pattern of horizontal zoom UIs, where the left sides represents zoom-out and the right represents zoom-in.

5.2 Z-Axis Interactions for Window Hierarchy

As described in Section 2.1, the window hierarchy increases from the lowest level of content to pane(tab), window, and virtual screen. Like Pressture [45], which mapped stronger pressure to higher hierarchy levels, OtM can map higher lift gestures to higher-level

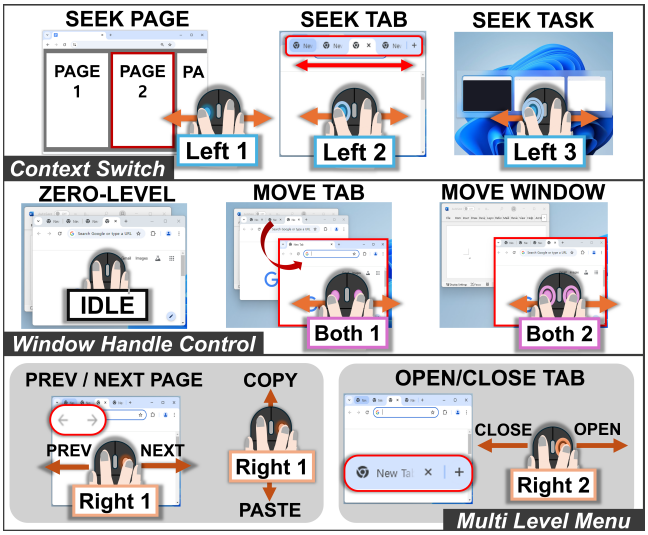

Figure 9: Z-axis interaction scenario for window hierarchy: Context switch, Window handle control, Multi level menu. For multi-level menus, a pie-shaped UI can be used to provide guidance to the user.

hierarchy interaction. Scenarios for window hierarchy are depicted in Figure 9.

Context switching. We map index finger-liftings to implement various functions in a browser across hierarchical levels. Finger-lifting at level 1 perform content-level interactions, while level 2 and 3 lifts handle tab and window interactions. Users can move the mouse left or right to navigate pages, tabs, or windows while lifting a finger to the corresponding level.

Window handle control. Windows are resized or repositioned using handles, and tabs can be separated or merged. We assigned both-finger lifting to control window handles. As seen in Figure 9, lifting both fingers to the level 1 allows free tab movement. At level 2, users can move window handles that contain several tabs, which operate at a higher level than tabs.

Multi-level menu. The middle-finger lifting can be mapped to right-click menu options. A first-level lift performs the document's right-click menu, allowing left-right strokes for moving to prev/next pages and up-down for copy-paste. A second-level lift triggers the tab's menu, with left-right strokes to open or close tabs. For interactions requiring different strokes with the same finger-lifting, as illustrated in Figure 9, displaying a pie-menu UI can inform users about the required stroke distance and the corresponding functionalities.

5.3 Z-Axis Interactions for Layer-Based Interfaces

Layer panel control In software like *Photoshop*, OtM can facilitate layer panel control (Figure 10). Raising the index finger to level 1 activates layer seeking, while lifting it to level 2 moves to a higher layer, and lowering it back to level 1 moves to a lower layer. Stroking the mouse to the right confirms the layer selection. Similarly, the

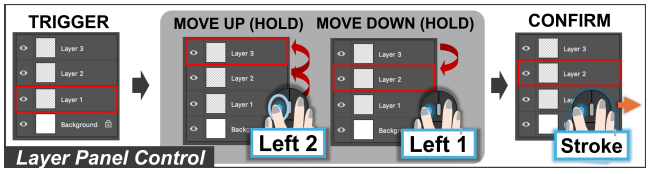

Figure 10: Z-axis interaction scenario for layer-based interface: Layer panel control on Photoshop.

middle finger can be used to reorder the selected layer. This interaction is illustrated in Figure 10. However, repetitive manipulation over a fixed interval can pose challenges: if the interval is too short, precise control becomes difficult; if too long, traversing distant layers becomes time-consuming. Therefore, careful adjustments, such as implementing adaptive or optimal intervals are important.

3D selection. In 3D UIs, commonly used in VR, OtM can enable direct 3D object selection via the z-axis. At this point, the existing mouse movement, where horizontal plane movement is reflected in vertical display movement, can be applied by mapping the z-axis of the mouse to the 3D forward-backward movement from the user, or for more intuitive 3D control, the z-axis can be mapped to the up-down direction in 3D space.

5.4 Z-Axis Interactions for Zooming Interfaces

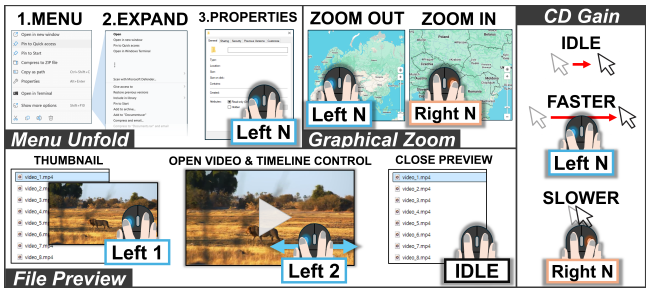

Figure 11: Z-axis interaction scenarios for zooming interface: Menu unfold, File preview, Graphical zoom, CD gain. Level 'N' denotes that manipulations are performed incrementally in accordance with levels 1, 2, and 3.

The zooming interface exemplifies z-axis interaction, with previous studies exploring graphical zoom [14, 42], semantic zoom [14, 42, 53], and input gain adjustment [14, 23]. Examples of zooming interface using *OtMouse* are shown in Figure 11.

Menu unfold. OtM enables users to expand information in file explorers. Moving the cursor to an icon and lifting the finger to level 1 opens the menu, level 2 expands it, and level 3 opens the *Properties* menu for detailed information.

Graphical zoom. In map applications, users can use OtM to zoom graphically. The index finger is used to zoom out, while the middle finger is used to zoom in. It also allows users to left-click navigation while zooming in. Users can adjust each finger's lift height to zoom in and out through multiple levels. This is similar to the *parameter control* scenario for continuous input but differs in that it allows users to quickly adjust between specific discrete

levels (e.g., citywide, street, or building zoom scale), offering distinct preset options for rapid control.

File preview. Using OtM, users can perform multi-level access to file icons in file systems. For images and documents, lifting the index finger to level 1 can open an enlarged preview. For videos, lifting the finger up to level 2 and moving left or right can allow users to navigate timeline. Lowering the finger down closes previews.

CD gain. OtM can adjust cursor's CD gain for multi-monitor setups or large displays. Lifting the index finger increases sensitivity for faster movement, while lifting the middle finger decreases it for precise control. Such CD gain manipulation does not modify the baseline gain of the mouse but can be effectively utilized in temporary and repetitive tasks that require quick movement across a wide monitor or precise cursor control.

Media control. In a media player, OtM allows users to control the content dynamically. Lifting the index finger to level 1 with left/right strokes adjusts the timeline by 1 second, level 2 by 5 seconds, and level 3 switches videos. Similarly, up/down strokes at level 1 adjust the volume by 5 units, level 2 by 10, and level 3 sets maximum or mute.

6 User Study 2: Usability Evalutaion on Z-Axis Interaction using OtM

This experiment compared the usability of OtM with a conventional mouse interface across three tasks derived from the representative scenarios in Section 5. Sixteen participants (5 females, 11 males; age: M = 23.3 years, SD = 2.8), all right-handed mouse users, were recruited for the evaluation. The study was conducted in a desktop setup with a 27-inch monitor, keyboard, and *OtMouse* and took approximately 90 minutes per participant.

Participants completed three tasks under two conditions: OtM (**OtM**) and conventional mouse (**base**). The tasks were performed sequentially for each condition, with three blocks per task. The order of conditions was counterbalanced. In the **OtM** condition, participants were briefed on the gestures mapped to each task and given 5 minutes for practice and familiarization with the gestures. In the **base** condition, they reviewed the GUI layout used for each task in advance.

After completing three blocks of each task, participants filled out a System Usability Scale (SUS) [6] survey of the completed tasks. Following all tasks under one condition, the NASA Task Load Index (NASA-TLX) survey [22] was administered. This procedure aimed to evaluate the usability of each scenario and assess the overall task load when various interactions were employed. At the end of the experiment, participants took part in a brief interview to share their scenario preferences. Movement data and sensor values from *OtMouse* were recorded during tasks for CT analysis.

The following section describes the three tasks and the gestures and mouse actions used for each condition. Then we compare the SUS scores, CT for each task, and NASA-TLX scores across conditions. This analysis evaluates the usability of the OtM interface for each task and assesses the overall task load experienced when using *OtMouse*. Finally, we summarize the interview responses.

6.1 Task 1: Browse and Task Switch

This task depicts a situation where inter-hierarchical navigations occur while multitasking. The task was performed on the *Chrome* browser and *MS Word*. At the beginning of the task, participants were presented with a maximized browser with seven tabs and an empty Word document hidden behind tabs. Each Chrome tab had five sentences containing a random number below 100 with a randomly generated sentence (i.e., *"35: Sample Sentence."*). At first, participants had to compare the first sentence in each tab and find the largest number and copy the sentence next to the number whenever they found a larger one. After finding the sentence with the largest number, they switched to Word and pasted the copied sentence. They then returned to the browser to compare the second sentence in each tab and repeated navigating, comparing, copying, and pasting actions until the last comparison was completed. To perform this task in **OtM** condition, we applied gestures described in Section 5.1 and illustrated in Figure 9 with some variations. We mapped the 'Seek Task' function to 'both level 3 lift,' instead of 'left level 3 lift.' And the action of each gesture was displayed with the pie-menu format according to their finger-lifting. In the **base** condition, left and right clicks, tab pane on Chrome, and Taskbar at the bottom of the screen were used.

6.2 Task 2: Video Search and Menu Select

This task consists of exploring the contents of video files in a file management program and hiding files that satisfy specific conditions. The task was performed in *Windows Explorer* and used *Windows Media Player* for video playback in **base** condition. At the beginning of the task, participants were given a directory containing twenty videos named '1.mp4' to '20.mp4' in the '*Details*(listed)' view. Each video was a five-minute wildlife documentary, and a random number between 0 and 9 was displayed for 10 seconds at once between 1 and 4 minutes randomly. They were instructed to open and navigate each video one-by-one, and if they found a video displaying the number 3 during playback, they were instructed to close the player and hide that video file in the *Explorer*. The number of videos appearing the number 3 ranged from two to four out of twenty and was presented pseudo-randomly for each trial. Videos were played in full-screen mode in all conditions, and the *Explorer* was presented in the center of the screen with a constant size.

For task 2, we mapped the 'File Preview' and 'Menu Unfold' scenarios in Section 5.3 (Figure 11) for **OtM** condition. In **base** condition, participants used mouse clicks and dragging the cursor along the playback bar to perform the task.

6.3 Task 3: Find Places on the Map

This task involves a detailed exploration in an online map through *Google Maps* in *Chrome*. It simulates a user repeatedly zooming in and out of the UI to find specific targets. At the beginning of the task, participants were shown Maps of specific cities. These cities were grouped and selected based on areas: Group 1 of 'San Francisco,' 'Florence,' and 'Barcelona;' Group 2 of 'Lisbon,' 'Vancouver,' and 'Copenhagen.' The order within the group was fixed, but the group was counterbalanced across conditions. Each city had six marked places, and participants had to find the nearest restaurant to each spot. The answer for the nearest restaurant was withheld; instead,

they were asked to proceed to the next place based on their selection to evaluate search speed and accuracy.

In **OtM** condition, participants used the 'Graphical Zoom' scenario described in section 5.4 (Figure 11). To zoom in on the map, participants lifted their middle finger, navigating up to three levels, and then moved the map by left-clicking and dragging. To zoom out, they lifted their index finger, allowing zooming out up to three levels to view the entire city at once. In **base** condition, they used the scroll wheel to zoom in and out of the map.

6.4 Result and Discussion

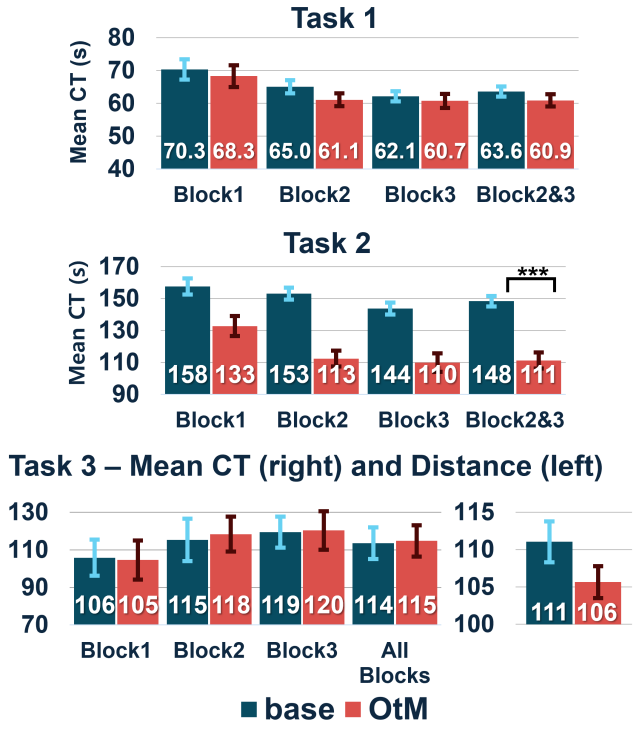

Figure 12: The results of the CT measurement according to block for each task and calculated mean distance for Task 3. Error bars show standard errors, and asterisks indicate signifcant diferences (*: p < .001)**

6.4.1 Task Completion Time. The results of CT measurement are shown in Figure 12. CT for each task was measured from the moment participants moved the mouse to moment they completed it. For Task 1, a one-way RM-ANOVA was conducted to check the learning effect across Blocks. Both conditions showed significant improvements in CT (base: F (2, 30) = 7.147, p <. 003; OtM: F (2, 30) = 6.423, p < .001) during repetitions. Thus, we performed a paired samples T-test between conditions using the mean value of Block 2 and 3, and no significant CT difference was found. In Task 2, one-way RM-ANOVA results also showed significant CT improvement during repetitions (base: F (2, 30) = 9.118, p < .001; OtM, corrected using Greenhouse-Geisser estimates: F (1.471, 22.070) = 206.193, p < .001). A paired samples T-test showed that **OtM** had a significantly improved CT of about 22% compared to the **base** condition (t(15) = 7.908, p < .001). Since Task 3 explored different cities for each block, a paired samples T-test was performed between conditions using all Block values. There was no significant difference in CT between **base** and **OtM** conditions. Through this, the OtM interface, mapped for our scenarios, demonstrated significant improvements in efficiency compared to the existing approach in tasks such as opening and navigating files or expanding menus. This improvement is attributed to the mouse movements required for each scenario: while Task 1 and Task 3 involved cursor movements such as dragging sentences or swiping maps, which were present in both **OtM** and **base**, Task 2 primarily resolved cursor movements through lift gestures, making a remarkable difference. The average distance from the place to the designated restaurant, measured in Task 3, was about 5 meters closer in the **OtM** condition than in the **base**, but it was not statistically significant.

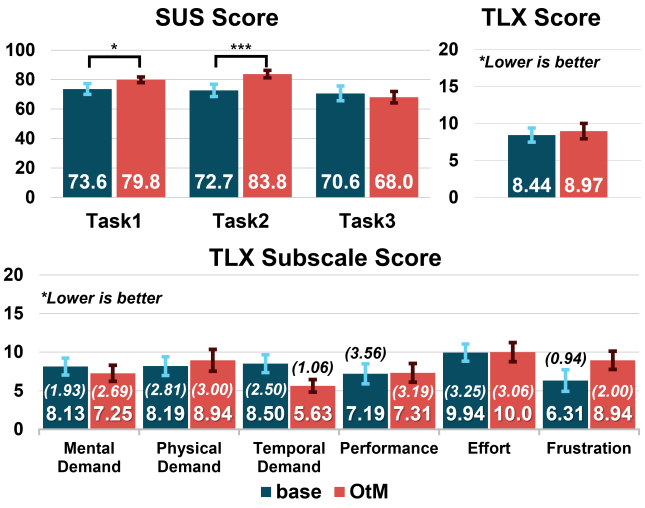

Figure 13: The scores of SUS questionnaire for each task, and mean TLX score for each subscale and total score with weight. Error bars show standard errors, values in parenthesis are the average weight, and asterisks indicate significant differences (*: p < .05, ***: p < .001)

6.4.2 SUS and TLX Result. The SUS scores for each task under both conditions are shown in Figure 13. For Tasks 1 and 3, a paired-samples T-Test was used to compare the SUS scores between conditions. For Task 2, the Shapiro-Wilk test revealed a violation of normality, so a Wilcoxon signed-rank Test was conducted. In Task 1, a significant usability improvement of 6.2 points was observed in **OtM** (t(15) = 2.259, p < .039). Task 2 also showed a significant advantage for **OtM** (W = 84, p = .008) by 11.1 points. For the subscale scores of TLX, none of these were statistically significant.

6.5 Qualitative Feedback

The results reveal that for Task 1 and Task 2, both average CT and SUS scores improved, with SUS scores showing significant differences. However, for Task 3, there was no improvement in either

Condition	Task1	Task2	Task3
base	**Total (2)** *- Familiarity* (P2, P13)	**Total (3)** *- Difficulties in holding the finger lifted up in **OtM*** (P2, P6, P14)	**Total (10)** *- Zoom fixed after adjustment* (P7, P10) *- Precise zoom control* (P2, P4, P6-10, P13-15)
OtM	**Total (14)** *- Reduction in cursor movement* (P1, P3, P5, P8-9, P12, P14-16) *- Mitigate the risk of wrong selection* (P3-4, P6-7, P10-11)	**Total (13)** *- Direct video control after file selection* (P1, P3-5, P8, P10, P16) *- Reduced burden of search/click* (P1, P5, P7, P9, P11-12) *- Quick transition to other file* (P13, P15)	**Total (6)** *- Intuitive operation* (P1, P5, P11-12) *- Fast scale conversion* (P3, P16)

Table 1: The user preference response results to the question *'Which of the two conditions did you prefer for each task?'* and the grouped reason for their choice.

CT or SUS. This trend was reflected in the post-task preference interview, as shown in Table 1.

In Task 1, all 16 participants found the OtM functions intuitive, regardless of their preferred method. Participants who favored **OtM** highlighted its benefits in reducing mouse movement and clicking, as well as mitigating errors like accidentally closing tabs or clicking incorrect icons. P4 noted, *"While using the traditional method, I accidentally closed a tab due to a cursor misstep, which was embarrassing. But, this mouse (OtMouse) reduced such mistakes."* Those who preferred the conventional method cited familiarity as their primary reason. These findings suggest that OtM improved usability for most users by simplifying mouse operations.

In Task 2, all participants noted that **OtM** was intuitive. Those who preferred **OtM** cited the ability to control videos directly after selecting a file, reduced effort in menu navigation, and quick transitions to the next file. P15 remarked, *"It was good that when I closed the video, I could check the next file by lowering the cursor slightly."* Conversely, participants preferred the **base** mentioned difficulties with moving the mouse while lifting a finger. P2 said, *"When I lifted my finger and as I moved the mouse, my finger slowly went down, and the player closed."* In summary, **OtM** reduced unnecessary mouse movements and simplified navigation in file preview and expandable menu, improving usability and CT for most users. However, some users found finger-lifting to be a physical burden.

In Task 3, some participants experienced confusion between the functions assigned to the index and middle fingers. Participants who preferred **OtM** stated its intuitive operation and fast scale conversion. Those favoring the **base** highlighted difficulties with precise control and some additionally mentioned challenges with cursor dragging while zooming in. P10 remarked, *"I wish there were more control stages, and I found it difficult to lift my middle finger and click-drag with my index finger."* In short, participants tended to prefer using the scroll wheel over **OtM** z-axis mapping in graphical zoom scenarios, as shown in Table 1. This preference likely stems from a need for fine-grained control and the ability to maintain the zoom scale after selecting it.

Throughout the experiment, participants were familiar with the traditional mouse system but less with OtM. Despite the five minutes of practice and training during the study, participants still found the new mouse operation unfamiliar, which is somewhat anticipated. A future study with long-term exposure to OtM interface could provide more precise usability insights.

7 General Discussion and Future Work

7.1 OtM Interface Enhanced Z-Axis Interaction

We evaluated the OtM interaction with three representative tasks. We compared CT, SUS scores, and the NASA-TLX scores between the OtM and conventional mouse conditions and collected interview responses. The results showed that the TLX scores were almost the same across conditions and that the OtM condition showed a significant improvement in SUS scores compared to the conventional mouse condition for Tasks 1 and 2 and a significant improvement in CT for Task 2. Participants who preferred OtM (Task 1: 14 of 16, Task 2: 13 of 16) reported that it reduced unnecessary movements and eased the operational burden. This indicates that OtM can improve the current GUIs by allowing intuitive and efficient navigation in the UI z-axis.

7.2 Comparison with Other Input Options

In User Study 1, we measured the performance of finger-lifting operations. The operations were performed with CT of 0.47 seconds and ER of 1.3% in two-level discrimination and CT of 0.63 seconds and ER of 2.5% in three-level discrimination. In Cechanowicz's study [12], they evaluated level selection performance using pressure-sensitive buttons on a mouse for four or more discrete levels. Although a direct comparison is difficult, our three-level distinction demonstrates a performance comparable to their four-level distinction performance (0.9s CT, 9%ER). Additionally, *OtMouse* shows higher gesture accuracy than Presstures [45], three-level pressure gestures on a touchpad (28.5% ER). These comparisons

may suggest that the performance of *OtMouse*'s finger-lifting operations is competitive over force-based z-axis operations.

We did not compare the usability of OtMouse with a multi-button mouse. Instead, we discuss expected differences between the two cases here. As the number of commands to support increases, multi-button mice may not scale effectively; that is, multi-button mice may need an increased number of buttons, as the number of required commands increases. As noted by Cechanowicz [12], a large number of buttons on the mouse complicates memorization and necessitates hand repositioning. In contrast, the OtMouse may expand its input space dynamically; that is, it may increase the number of input levels, though at the expense of precision, thereby accommodating a broader range of commands. For example, in User Study 2, we actively adjusted the two- and three-level discrimination gestures according to the required lifting levels. Although discrimination levels of four or more steps may also be applied as needed, their usability warrants further validation. Another expected issue with multi-button mice is that all of the additional buttons should be manipulated by the thumb alone. In contrast, the OtMouse can assign different functions to different fingers or finger combinations, such as using the index finger for selection and the middle finger for menu options, enabling a better operation-function mapping. Finally, the OtMouse's vertical finger-lifting action may more naturally correspond to z-axis GUI navigation than horizontal thumb-side buttons of multi-button mice.

In this study, we focused on one-handed interaction with the mouse. Nevertheless, we may compare the performance of finger-lifting operations with that of keyboard shortcuts, which can be entered with a non-preferred hand in two-handed interactions. According to the keystroke-level model [9], a keystroke requires 0.2 seconds. A keyboard shortcut is often entered by pressing two keys (a modifier and a symbol key) in combination, so we may estimate the input time for a keyboard shortcut to be approximately 0.4 seconds. This suggests that a keyboard shortcut may be slightly faster than a 2-level lifting operation. However, the *OtMouse*'s finger-lifting, when combined with 'directional' strokes on the mouse plane, can facilitate an intuitive command mapping and may offset the speed benefit of keyboard shortcuts.

7.3 Limitation of the Current Interaction Design

For Task 3 in User Study 2, there were no significant improvements in CT or in SUS, and most participants (10 of 16) preferred the traditional scroll wheel method. All participants who favored the wheel method mentioned the lack of operations for fine zoom-level adjustment, and difficulty in performing drag actions while maintaining a finger-lifting operation in OtM interface. Regardless of preference, four participants stated that the actions mapped to each finger (index: zoom-out, middle: zoom-in) were sometimes confusing. These suggest that the interaction design for Task 3 is not yet mature and should be improved. For example, allowing more zoom levels beyond three or providing operations for continuous adjustment of zoom scale may improve the performance of Task 3. The burden of maintaining finger-lifting while dragging could be relieved by ensuring that the adjusted zoom scale remains 'locked,' even after the finger is lowered, which is a common strategy in previous force-based interface research [24]. The problem of confusing finger mapping may be solved by using only one finger for both zooming in and out; the height of the finger may be mapped to the current zoom level. In this case, users may need a switching method to select between zooming in or out and continuing it. We may adopt a velocity-based switching method, that adjusts the zoom scale through slow finger height adjustment while disregarding rapid finger motion, thereby enabling a broader zoom scale manipulations within a confined lifting range.

There is room for improvement also for Task 2. Three participants in Task 2 preferred the traditional method due to the burden of moving the cursor while maintaining finger-lifting, and mentioned that their finger unintentionally lowered, causing the video preview to end. This issue may also be addressed by a 'lock' mechanism [24] that we suggested above for Task 3. For instance, users may move the cursor a little while the video preview is active to lock the preview mode and move the cursor further to unlock the mode and close the preview. Users will not need to maintain their finger over the mouse once the mode is locked.

To fully harness the advantages of the OtM interface, further studies to refine the design details, including optimal finger mapping and techniques for locking an input mode, will be necessary.

7.4 Future Enhancements for the OtMouse

There is room for improvements in the design of *OtMouse*. One participant experienced a temporary failure in lift detection due to holding the mouse at an angle, causing his finger to move out of the ToF sensor's range. This issue may be resolved by using ToF sensors with a broader sensing range. In addition, three participants said the current mouse form is slightly burdensome for finger-lifting. This may mean that finger-lifting while holding the mouse was uncomfortable for them. Improving interaction design may help, but modifying the mouse's shape to reduce the burden of finger-lifting may be a more fundamental solution. Lifting the finger strains the wrist, like bending the wrist upward. This can compress the median nerve in the carpal tunnel, potentially causing pain or numbness in the fingers for some users [7, 43, 44]. Existing mice often adopt asymmetric angled or vertical designs to alleviate wrist pressure [13, 19, 27, 39]. Therefore, the *OtMouse* could consider a design that reduces the burden of the lift motion. However, a tilted mouse shape may alter the direction of the finger-lifting, potentially aggravating the vertical z-axis metaphor. Further design iteration is required to address the potential ergonomic issues of the lift operations and measure changes in finger fatigue during prolonged use of the *OtMouse*.

8 Conclusion

We presented *OtMouse*, a novel z-axis input mouse that supports finger-lifting operations. We evaluated the usability of these operations for different finger-lifting levels, and assessed the usability and efficiency of the OtM interface with GUI tasks involving navigation along the z-axis. The results of the main user study showed that the OtM interface significantly improved usability and received high user preference scores in scenarios requiring cursor movement within inter-hierarchy interactions, such as task switching

and semantic zooming interactions. Additionally, participant interviews provided valuable insights into possible future improvements for both the OtM interface and the *OtMouse* device. These results indicate that the OtMouse and the OtM interface have the potential to enhance the user experience of the future desktop GUIs.

Acknowledgments

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2024-RS-2024-00436398) supervised by the IITP(Institute for Information & Communications Technology Planning & Evaluation)

We also thank Yoonha Bahng and Natapat Klaewkla for their help in initial exploration of the over the mouse interaction.

References

[1] Adobe. 2023. *Nondestructive editing*. Retrieved Sep. 2024 from https://helpx.adobe.com/photoshop/using/nondestructive-editing.html

[2] Michelle Annett, Tovi Grossman, Daniel Wigdor, and George Fitzmaurice. 2011. Medusa: a proximity-aware multi-touch tabletop. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Santa Barbara, California, USA) *(UIST '11)*. Association for Computing Machinery, New York, NY, USA, 337–346. doi:10.1145/2047196.2047240

[3] Ravin Balakrishnan, Thomas Baudel, Gordon Kurtenbach, and George Fitzmaurice. 1997. The Rockin'Mouse: integral 3D manipulation on a plane. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems* (Atlanta, Georgia, USA) *(CHI '97)*. Association for Computing Machinery, New York, NY, USA, 311–318. doi:10.1145/258549.258778

[4] Gábor Blaskó and Steven Feiner. 2004. Single-handed interaction techniques for multiple pressure-sensitive strips. In *CHI '04 Extended Abstracts on Human Factors in Computing Systems* (Vienna, Austria) *(CHI EA '04)*. Association for Computing Machinery, New York, NY, USA, 1461–1464. doi:10.1145/985921.986090

[5] Peter Bogunovich and Dario Salvucci. 2011. The effects of time constraints on user behavior for deferrable interruptions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 3123–3126. doi:10.1145/1978942.1979404

[6] John Brooke. 1995. SUS: A quick and dirty usability scale. *Usability Eval. Ind.* 189 (11 1995).

[7] Claire Burton, Linda S Chesterton, and Graham Davenport. 2014. Diagnosing and managing carpal tunnel syndrome in primary care. *British Journal of General Practice* 64, 622 (2014), 262–263. doi:10.3399/bjgp14X679903 arXiv:https://bjgp.org/content/64/622/262.full.pdf

[8] David M. Cades, Patrick E. McKnight, David G. Kidd, Eden B. King, and Deborah A. Boehm-Davis. 2010. Factors Affecting Interrupted Task Performance: Effects of Adaptability, Impulsivity and Intelligence. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 54, 4 (2010), 458–462. doi:10.1177/154193121005400439

[9] Stuart K. Card, Thomas P. Moran, and Allen Newell. 1980. The keystroke-level model for user performance time with interactive systems. *Commun. ACM* 23, 7 (July 1980), 396–410. doi:10.1145/358886.358895

[10] Géry Casiez, Nicolas Roussel, and Daniel Vogel. 2012. 1 € filter: a simple speed-based low-pass filter for noisy input in interactive systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Austin, Texas, USA) *(CHI '12)*. Association for Computing Machinery, New York, NY, USA, 2527–2530. doi:10.1145/2207676.2208639

[11] Géry Casiez, Daniel Vogel, Ravin Balakrishnan, and Andy Cockburn. 2008. The Impact of Control-Display Gain on User Performance in Pointing Tasks. *Human-computer Interaction* 23 (07 2008), 215–250. doi:10.1080/07370020802278163

[12] Jared Cechanowicz, Pourang Irani, and Sriram Subramanian. 2007. Augmenting the mouse with pressure sensitive input *(CHI '07)*. Association for Computing Machinery, New York, NY, USA, 1385–1394. doi:10.1145/1240624.1240835

[13] Han-Ming Chen and Chun-Tong Leung. 2007. The effect on forearm and shoulder muscle activity in using different slanted computer mice. *Clin Biomech (Bristol, Avon)* 22, 5 (March 2007), 518–523.

[14] Xiang 'Anthony' Chen, Julia Schwarz, Chris Harrison, Jennifer Mankoff, and Scott E. Hudson. 2014. Air+touch: interweaving touch & in-air gestures. *Proceedings of the 27th annual ACM symposium on User interface software and technology* (2014). https://api.semanticscholar.org/CorpusID:14474047

[15] Mehmet Celal Dasiyici. 2008. Multi-Scale Cursor: Optimizing Mouse Interaction for Large Personal Workspaces. https://api.semanticscholar.org/CorpusID:59737946

[16] Clifton Forlines, Chia Shen, and Bill Buxton. 2005. Glimpse: a novel input model for multi-level devices *(CHI EA '05)*. Association for Computing Machinery, New York, NY, USA, 1375–1378. doi:10.1145/1056808.1056920

[17] J. Franz, A. Menin, and L. Nedel. 2016. Lossless Multitasking: Using 3D Gestures Embedded in Mouse Devices. In *2016 XVIII Symposium on Virtual and Augmented Reality (SVR)*. 109–116. doi:10.1109/SVR.2016.27

[18] S. Frees and G.D. Kessler. 2005. Precise and rapid interaction through scaled manipulation in immersive virtual environments. In *IEEE Proceedings. VR 2005. Virtual Reality, 2005.* 99–106. doi:10.1109/VR.2005.1492759

[19] Ewa Gustafsson and Mats Hagberg. 2003. Computer mouse use in two different hand positions: exposure, comfort, exertion and productivity. *Appl Ergon* 34, 2 (March 2003), 107–113.

[20] Winfried Hacker. 1988. *Computerization versus computer aided mental*. North-Holland Publishing Co., NLD, 115–130.

[21] Roger A Hart and Gary T Moore. 1973. The Development of Spatial Cognition: A Review. In *Image & environment: Cognitive mapping and spatial behavior*. AldineTransaction, New Brunswick, NJ, US, 246–288.

[22] Sandra G. Hart and Lowell E. Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In *Human Mental Workload*, Peter A. Hancock and Najmedin Meshkati (Eds.). Advances in Psychology, Vol. 52. North-Holland, 139–183. doi:10.1016/S0166-4115(08)62386-9

[23] Seongkook Heo, Jaehyun Han, and Geehyuk Lee. 2015. Designing for Hover- and Force-Enriched Touch Interaction. In *Computer-Human Interaction. Cognitive Effects of Spatial Interaction, Learning, and Ability*, Theodor Wyeld, Paul Calder, and Haifeng Shen (Eds.). Springer International Publishing, Cham, 68–87.

[24] Seongkook Heo and Geehyuk Lee. 2012. ForceDrag: using pressure as a touch input modifier. In *Proceedings of the 24th Australian Computer-Human Interaction Conference* (Melbourne, Australia) *(OzCHI '12)*. Association for Computing Machinery, New York, NY, USA, 204–207. doi:10.1145/2414536.2414572

[25] Otmar Hilliges, Shahram Izadi, Andrew D. Wilson, Steve Hodges, Armando Garcia-Mendoza, and Andreas Butz. 2009. Interactions in the air: adding further depth to interactive tabletops. In *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology* (Victoria, BC, Canada) *(UIST '09)*. Association for Computing Machinery, New York, NY, USA, 139–148. doi:10.1145/1622176.1622203

[26] Ken Hinckley, Mike Sinclair, Erik Hanson, Richard Szeliski, and Matt Conway. 1999. The VideoMouse: a camera-based multi-degree-of-freedom input device *(UIST '99)*. Association for Computing Machinery, New York, NY, USA, 103–112. doi:10.1145/320719.322591

[27] Annemieke Houwink, Karen M Oude Hengel, Dan Odell, and Jack T Dennerlein. 2009. Providing training enhances the biomechanical improvements of an alternative computer mouse design. *Hum Factors* 51, 1 (Feb. 2009), 46–55.

[28] Logitech Inc. 2023. *Logitech G600 MMO Gaming Mouse*. Retrieved Sep. 2024 from https://www.logitechg.com/en-us/products/gaming-mice/g600-mmo-gaming-mouse

[29] Razer Inc. 2022. *RAZER Naga V2 PRO*. Retrieved Sep. 2024 from https://www.razer.com/gaming-mice/razer-naga-v2-pro

[30] Inseong Lee Jinkyu Jang and Jinwoo Kim. 2017. The Effects of Hover Interface on Users' Behavioral Multitasking Intention. *International Journal of Human–Computer Interaction* 33, 7 (2017), 537–548. doi:10.1080/10447318.2016.1256038

[31] Seoktae Kim, Hyunjung Kim, Boram Lee, Tek-Jin Nam, and Woohun Lee. 2008. Inflatable mouse: volume-adjustable mouse with air-pressure-sensitive input and haptic feedback. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Florence, Italy) *(CHI '08)*. Association for Computing Machinery, New York, NY, USA, 211–224. doi:10.1145/1357054.1357090

[32] Douglas C. Maynard and Milton D. Hakel. 1997. Effects of Objective and Subjective Task Complexity on Performance. *Human Performance* 10, 4 (1997), 303–330. doi:10.1207/s15327043hup1004_1

[33] D E Meyer and D E Kieras. 1997. A computational theory of executive cognitive processes and multiple-task performance: Part 1. Basic mechanisms. *Psychol Rev* 104, 1 (Jan. 1997), 3–65.

[34] Sachi Mizobuchi, Shinya Terasaki, Turo Keski-Jaskari, Jari Nousiainen, Matti Ryynanen, and Miika Silfverberg. 2005. Making an impression: force-controlled pen input for handheld devices. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems* (Portland, OR, USA) *(CHI EA '05)*. Association for Computing Machinery, New York, NY, USA, 1661–1664. doi:10.1145/1056808.1056991

[35] Weimin Mou, Timothy P McNamara, Björn Rump, and Chengli Xiao. 2006. Roles of egocentric and allocentric spatial representations in locomotion and reorientation. *J Exp Psychol Learn Mem Cogn* 32, 6 (Nov. 2006), 1274–1290.

[36] Miguel A. Nacenta, Samer Sallam, Bernard Champoux, Sriram Subramanian, and Carl Gutwin. 2006. Perspective cursor: perspective-based interaction for multi-display environments *(CHI '06)*. Association for Computing Machinery, New York, NY, USA, 289–298. doi:10.1145/1124772.1124817

[37] Mathieu Nancel, Emmanuel Pietriga, Olivier Chapuis, and Michel Beaudouin-Lafon. 2015. Mid-Air Pointing on Ultra-Walls. *ACM Trans. Comput.-Hum. Interact.* 22, 5, Article 21 (aug 2015), 62 pages. doi:10.1145/2766448

[38] Gary Perelman, Marcos Serrano, Mathieu Raynal, Celia Picard, Mustapha Derras, and Emmanuel Dubois. 2015. The Roly-Poly Mouse: Designing a Rolling Input Device Unifying 2D and 3D Interaction. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 327–336. doi:10.1145/2702123.2702244

[39] Paulo R V Quemelo and Edgar Ramos Vieira. 2013. Biomechanics and performance when using a standard and a vertical computer mouse. *Ergonomics* 56, 8 (June 2013), 1336–1344.

[40] Adrian Ramcharitar and Robert J. Teather. 2018. EZCursorVR: 2D Selection with Virtual Reality Head-Mounted Displays. In *Proceedings of the 44th Graphics Interface Conference* (Toronto, Canada) *(GI '18)*. Canadian Human-Computer Communications Society, Waterloo, CAN, 123–130. doi:10.20380/GI2018.17

[41] Gonzalo Ramos, Matthew Boulos, and Ravin Balakrishnan. 2004. Pressure widgets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vienna, Austria) *(CHI '04)*. Association for Computing Machinery, New York, NY, USA, 487–494. doi:10.1145/985692.985754

[42] Jun Rekimoto and Carsten Schwesig. 2006. PreSenseII: bi-directional touch and pressure sensing interactions with tactile feedback. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems* (Montréal, Québec, Canada) *(CHI EA '06)*. Association for Computing Machinery, New York, NY, USA, 1253–1258. doi:10.1145/1125451.1125685

[43] D Rempel, J M Bach, L Gordon, and Y So. 1998. Effects of forearm pronation/supination on carpal tunnel pressure. *J Hand Surg Am* 23, 1 (Jan. 1998), 38–42.

[44] D Rempel, L Dahlin, and G Lundborg. 1999. Pathophysiology of nerve compression syndromes: response of peripheral nerves to loading. *J Bone Joint Surg Am* 81, 11 (Nov. 1999), 1600–1610.

[45] Christian Rendl, Patrick Greindl, Kathrin Probst, Martin Behrens, and Michael Haller. 2014. Presstures: exploring pressure-sensitive multi-touch gestures on trackpads. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 431–434. doi:10.1145/2556288.2557146

[46] Simon Rogers, John Williamson, Craig Stewart, and Roderick Murray-Smith. 2011. AnglePose: robust, precise capacitive touch tracking via 3d orientation estimation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 2575–2584. doi:10.1145/1978942.1979318

[47] Kang Shi, Sriram Subramanian, and Pourang Irani. 2009. PressureMove: Pressure Input with Mouse Movement. In *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part II* (Uppsala, Sweden) *(INTERACT '09)*. Springer-Verlag, Berlin, Heidelberg, 25–39. doi:10.1007/978-3-642-03658-3_7

[48] Yoshiki Takeoka, Takashi Miyaki, and Jun Rekimoto. 2010. Z-touch: an infrastructure for 3d gesture interaction in the proximity of tabletop surfaces. In *ACM International Conference on Interactive Tabletops and Surfaces* (Saarbrücken, Germany) *(ITS '10)*. Association for Computing Machinery, New York, NY, USA, 91–94. doi:10.1145/1936652.1936668

[49] Nicolas Villar, Shahram Izadi, Dan Rosenfeld, Hrvoje Benko, John Helmes, Jonathan Westhues, Steve Hodges, Eyal Ofek, Alex Butler, Xiang Cao, and Billy Chen. 2009. Mouse 2.0: multi-touch meets the mouse. In *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology* (Victoria, BC, Canada) *(UIST '09)*. Association for Computing Machinery, New York, NY, USA, 33–42. doi:10.1145/1622176.1622184

[50] Nicolas Villar, Shahram Izadi, Dan Rosenfeld, Hrvoje Benko, John Helmes, Jonathan Westhues, Steve Hodges, Eyal Ofek, Alex Butler, Xiang Cao, and Billy Chen. 2009. Mouse 2.0: multi-touch meets the mouse. In *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology* (Victoria, BC, Canada) *(UIST '09)*. Association for Computing Machinery, New York, NY, USA, 33–42. doi:10.1145/1622176.1622184

[51] Colin Ware and Kathy Lowther. 1997. Selection using a one-eyed cursor in a fish tank VR environment. 4, 4 (dec 1997), 309–322. doi:10.1145/267135.267136

[52] Vadim Zaliva. 2012. USB HID device abstraction for HDTP user interfaces. U.S. Patent Application No. 13/356,578, Filed Jan. 23th, 2012, Issued Jul. 26th 2012.

[53] Robert Zeleznik, Timothy Miller, and Andrew Forsberg. 2001. Pop through mouse button interactions. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology* (Orlando, Florida) *(UIST '01)*. Association for Computing Machinery, New York, NY, USA, 195–196. doi:10.1145/502348.502384

[54] Qian Zhou, George Fitzmaurice, and Fraser Anderson. 2022. In-Depth Mouse: Integrating Desktop Mouse into Virtual Reality *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 354, 17 pages. doi:10.1145/3491102.3501884