HANDS-FREE USER INTERACTION FOR ACCESSIBLE COMPUTING

MEHEDI HASSAN

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 COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

MAY 2019

Abstract

Three experiments were conducted to compare hands-free and hands-on input methods for accessible computing. The first experiment compared hands-free and hands-on text-entry on a smart-phone. *EVA Facial Mouse*, an Android application, was used for facial tracking for the hands-free phase of the experiment. The second experiment used the Fitts' law two-dimensional task in ISO 9241-9 to evaluate hands-free and hands-on point-select tasks on a laptop computer. We used a facial-tracking software called *Camera Mouse* in combination with dwell-time selection. The third user study compared *Camera Mouse* with the keyboard and touchpad of a laptop to play a simple yet well known game: *Snake*. A case study was also conducted with a physically-challenged participant for the hands-free phase of the gaming experiment. Our key finding from the three experiments and the case study is that the hands-free methods are not yet as well-performing as the hands-on methods.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The concept of human-computer interaction (HCI) is no longer confined to only physical interaction. While actions such as hovering the cursor of a mouse or touch-pad with our hands, clicking a button, typing, and playing games with our fingers are common forms of interaction, performing such tasks without physical touch is an intriguing idea. This is interesting both for non-disabled users and also for physically-impaired users. Gregor et al. [9] discussed the digital divide that exist towards physically-impaired people when it comes to modern technology. They note that developers often inadvertently exclude disabled people from their target group. They argue that the availability of vast technical knowledge combined with legislative activity and frameworks for accessible design create an impression that this digital divide does not exist. But, it does.

Our work aims to make this divide smaller. Any user input that does not require direct physical touch is potentially useful for physically-challenged users: The goal is

accessible computing. We divide our approach into three branches of user interaction: text entry, point-select tasks, and gaming. These three branches address essential and recreational aspects of user interaction. As we focus on the hands-free experience, it is imperative to find the proper tools for hands-free user input.

1.2 Approaches

Facial tracking for user input on mobile devices has potential in the pursuit of accessible computing. In the past, computer input was primarily through keyboards and pointing devices such as a mouse. Recently, touchscreen devices have been overtaking the keyboard-equipped devices. Shneiderman [25, p. 94] notes, "a common pursuit with touch screens is developing visually appealing metaphors that react predictably". Interaction with a generic smart-phone with a touchscreen involves the user touching the screen and initiating actions that generate touch events. This process involves direct physical contact with the device. Alternatively, a smart-phone's built-in accelerometer allows input actions to be generated by rotating the device.

We are particularly interested in methods that do not require specialized hardware, such as eye trackers. Our focus is on methods that use inexpensive built-in cameras, either on a laptop's display or in a smartphone or tablet. Tracking a body position, perhaps on the head or face, is easier than tracking the movement of a user's eyes, which undergo rapid jumps known as saccades [17]. The smoother and more gradual movement of the head or face, combined with the ubiquity of front-facing cameras on today's laptops, tablets, and smartphones, presents a special opportunity for users with motor disabilities. Such users desire access to the same wildly popular devices as used by non-disabled users.

In this research, scenarios are explored where physical touch is compared with

tilt input and with a hands-free method that uses facial-tracking for providing user input. *EVA Facial Mouse*[1] and *Camera Mouse*[2] are two such applications which serve our purpose. We used *EVA Facial Mouse* for a text-entry based user study that requires facial tracking. We used *Camera Mouse* for another user study based on a 2-D Fitts' law task to evaluate point-select tasks with facial tracking.

We also conducted an experiment with a very simple yet well-known game: *Snake*. Since MIT student Steve Russel made the first interactive computer game, *Spacewar* in 1961 [13], computer games have evolved a long way. But we did not conduct our experiment with a game that requires expensive graphics equipment or intense user interaction. For our study, the game was played with two hands-on methods and a hands-free method with facial tracking. We have re-created a version of *Snake* for this experiment. Our version is named *Snake: Hands-Free Edition*. This is a *Windows*-platform game. It can be played with three different input methods: keyboard, touchpad, and *Camera Mouse*. The first two methods are hands-on and the third method is hands-free. By conducting this experiment, we explore the opportunities of accessible computing with hands-free input for gaming. As part of our third user study, we conducted a case study as well. The participant for this case study has mild cerebral palsy. Due to his physical condition, the case study was only conducted for the hands-free phase of our gaming experiment. We later compared the results of the case study with the results of the user study. We have summarized the three experiments mentioned above in Table 1.1.

[1]https://play.google.com/store/apps/details?id=com.crea$_s$*i.eviacam.servicehl* = *en*$_C$*A*

[2]http://www.cameramouse.org/

Table 1.1: Summary of experiments.

Exp	**Title**	**Focus**	**Hardware**	**Software**
1	Comparing Hands-free and Hands-on Text Entry Methods for Mobile Devices.	Text-entry	Samsung S8+ smart-phone	*Keyboard Tester*, *EVA Facial Mouse*, *GoStats*.
2	A Fitts' Law Evaluation of Hands-free and Hands-on Input on a Laptop Computer.	Point-select tasks	Asus X541U laptop	*GoFitts,* *GoStats*, *Camera Mouse*.
3	Evaluating Hands-on and Hands-free Input Methods for a Simple Games.	Gaming	Asus X541U laptop	Snake: Hands-free edition, *Camera Mouse*, *GoFitts*, *GoStats*.

1.3 Research Contribution

1.3.1 Hands-free Text-entry with Opti

In our first user study, we evaluated two keyboard layouts on three input methods with *EVA Facial Mouse*. The keyboard layouts were *Qwerty* and *Opti*. Although *Opti* has been evaluated before for touch interaction, we present the first evaluation of *Opti* with facial tracking. We also present user feedback regarding the facial tracking tool *EVA Facial Mouse*.

1.3.2 A Study on Hands-free Point-select Tasks

In our second user study, we compared hands-on and hands-free methods for pointing and selecting. We used different dwell-time values used in similar research. We also evaluated an up-to-date version of *Camera Mouse*. We intend to conduct a case study in this experiment, with a participant who has a motor-disability in his/her hands.

1.3.3 Evaluation of a Hands-free Method for a Famous Video Game

In our third and final user study, we evaluated a facial tracking system (*Camera Mouse*) for a recreational purpose: a game (*Snake: Hands-free Edition*). Our goal, once again, was accessible computing. We also conducted a case study with a impaired participant with *Camera Mouse* to play this game.

1.4 Outline of Thesis

We describe our software systems, implementations, and theories in Chapter 2 which is followed by a literature review in Chapter 3. After that, we elaborate on the methodology and results of each experiment in Chapters 4, 5, and 6. Finally, we conclude and offer future opportunities for research in Chapter 7.

Chapter 2

Software Systems, Implementations and Theories

In this chapter, we discuss the software systems that were used fort the three experiments. Some of these systems already existed and some were custom implemented. We also discuss relevant HCI theories in this chapter.

2.1 EVA Facial Mouse

EVA Facial Mouse, a facial-tracking system, was used for this research. It is available for Android devices as an application from the Google *Playstore*. Facial-tracking is added as a feature from the accessibility settings of an Android device. *EVA Facial Mouse* – hereafter EFM – improves the accessibility of the device by manipulating a pointer controlled by facial movements.

Upon enabling EFM on an Android device, the front-facing camera tracks a point on the user's face, such as the nose tip or the point between the eyebrows (see Figure

2.1). Any movement of that point works like the movement of a mouse in controlling an on-screen pointer. Dwelling on one position for a pre-determined time initiates an event like a touch event or mouse button-click.

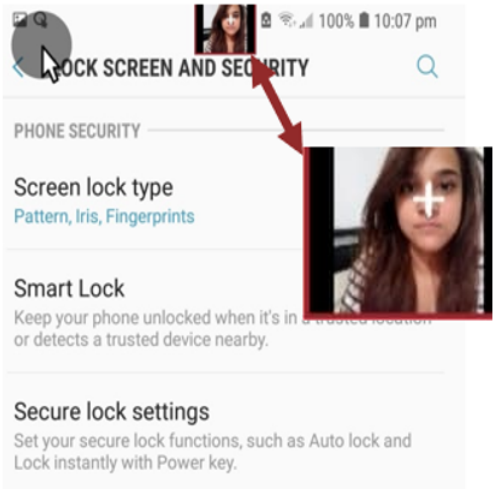


Figure 2.1: Facial-tracking by EFM.

Touch input is simple for everyone familiar with touch-based text-entry. But head movement and device tilt input methods are more involved. To put context into these two methods, some further elaboration on EFM is required. Upon enabling EFM in an Android device, the EFM mouse cursor appears on the device screen. The cursor can be moved in two ways.

2.1.1 Facial-tracking in Hands-free Mode

The cursor tracks a point on the user's head and makes that point a reference for movement. Hence, the device doesn't need to move. As seen in Figure 2.2, the EFM pointer can move in multiple directions based on the head movement of the user. The dark circle inside the red circle in Figure 2.2 represents a click-event. When

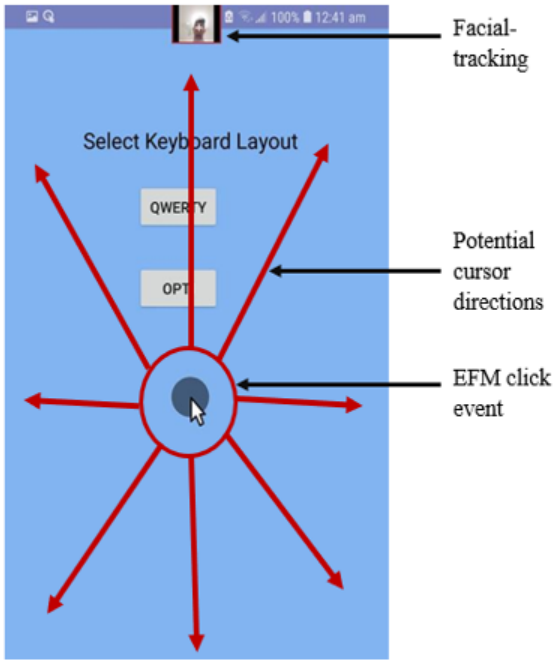


Figure 2.2: Facial-tracking by EFM in hands-free mode.

still, the cursor grows in size. Using dwell-time selection, a click event is generated after 1 second of no movement. Movement in 360 degrees is possible. These features were exploited during the hands-free phase of the experiment where the device was propped up on a laptop computer (see Figure 2.4).

2.1.2 Facial-tracking During Device Tilt

The device can be tilted sideways (see Figure 2.3-b, Figure 2.3-c), upwards (see Figure 2.3-d) or downwards (see Figure 2.3-a) for moving the EFM cursor. A steady head of the user is key for this. The movement of the pointer is always towards the opposite side of the tilt action. Bear in mind that the device accelerometer is not responsible for device tilt; the front-facing camera serves this purpose.

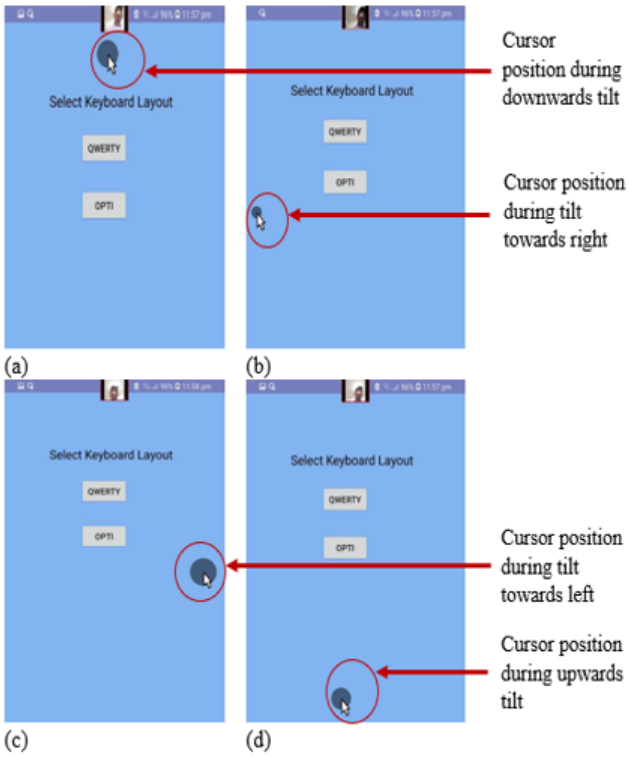


Figure 2.3: (a) EFM cursor position when the device is tilted downwards, (b) EFM cursor position when the device is tilted to the right, (c) EFM cursor position when the device is tilted to the left, (d) EFM cursor position when the device is tilted upwards.

2.2 Keyboard Tester

We developed an Android application called *Keyboard Tester*. The application provides conventional touch-based text-entry as well as facial-tracking-based text-entry with the aid of EFM. Our user study used *Keyboard Tester* to compare three input methods: touch, device tilt, and head movement. The first two methods are hands-on. The third, head movement, is hands-free. EFM is significant for this phase.

Figure 2.4: Device propped up on top of a laptop for the head movement (i.e., hands-free) method.

Keyboard Tester calculates the entry speed (in words per minute), error rate, and keystrokes per character (KSPC).

Soukoreff and MacKenzie [26] provide insight into measuring errors into text entry tasks. They introduced the minimum string distance (MSD) — originally credited to Levenshtein [14] for measuring error rates in text-entry evaluations. They propose using KSPC as a dependent measure. KSPC is 1 for simple text entry if all characters are correctly entered on the first attempt. KSPC rises from 1 for less skilled users when correcting mistakes during a text-entry task. MacKenzie [16] has noted various features of KSPC. He describes KSPC as the number of keystrokes required, on average, to generate a character of text for a given text entry technique in a given language. KSPC does not apply to keystrokes only. It can apply to stylus-based-input and hands-free input provided the entry method has the characteristics of a gesture stroke or tap.

2.3 The Opti Keyboard Layout

The *Keyboard Tester* software provides a choice of two keyboard layouts: *Qwerty* and *Opti*. *Qwerty* is the conventional keyboard layout. On the other hand, *Opti* is different in terms of letter arrangement (see Figure 2.5). *Opti* keys are organized in five rows compared to the four rows in a typical *Qwerty* keyboard layout. The *Opti* layout has four SPACE keys. This allows use of the SPACE key nearest to the key that has just been touched. Experiment data were collected on these two layouts for three input methods: touch, head movement (i.e., hands-free), and device tilt.

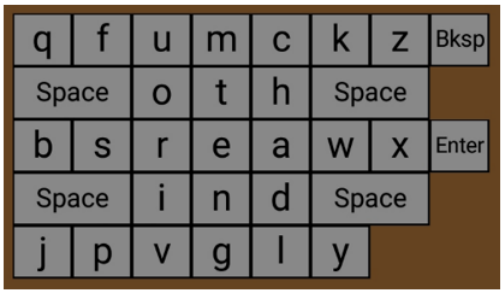


Figure 2.5: *Opti* keyboard layout as presented by MacKenzie [19, p. 275].

2.4 Camera Mouse

Camera Mouse is a facial tracking application that emulates a physical mouse. Upon enabling *Camera Mouse*, a point of the user's face is tracked. The preferable point of focus is usually the nose tip or the point between the eyebrows. If the user moves the tracked point of the face, the cursor moves on the screen accordingly. Figure 2.6 depicts how *Camera Mouse* tracks a point on a user's face.

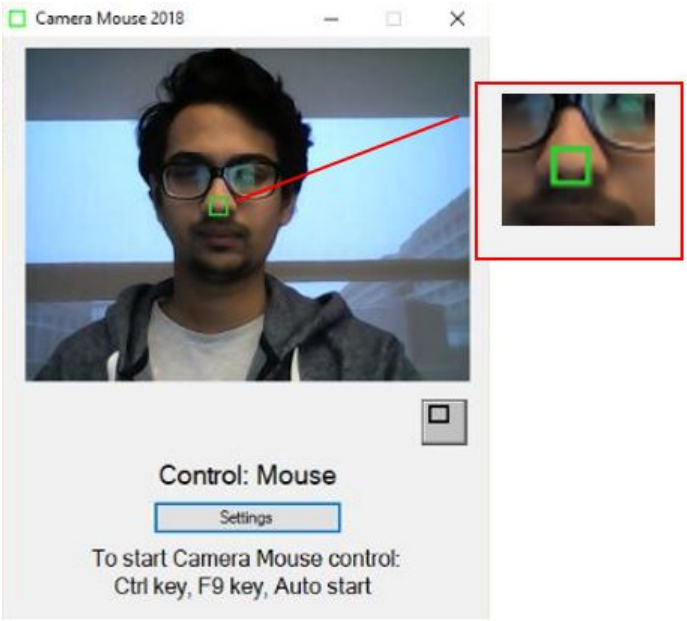


Figure 2.6: Face-tracking by *Camera Mouse*.

2.5 Evaluation Using Fitts' Law and ISO 9241-9

Fitts' law – first introduced in 1954 [6] – is a well-established protocol for evaluating target selection operations on computing systems [15, 2]. This is particularly true since the mid-1990s with the inclusion of Fitts' law testing in the ISO 9241-9 standard for evaluating non-keyboard input devices [27, 10, 11]. The most common ISO evaluation procedure uses a two-dimensional task with targets of width W arranged in a circle. Selections proceed in a sequence moving across and around the circle (see Figure 2.7). Each movement covers an amplitude A, the diameter of the layout circle. The movement time (MT, in seconds) is recorded for each trial and averaged over the sequence of trials.

The difficulty of each trial is quantified using an index of difficulty (ID, in bits) and is calculated from A and W as

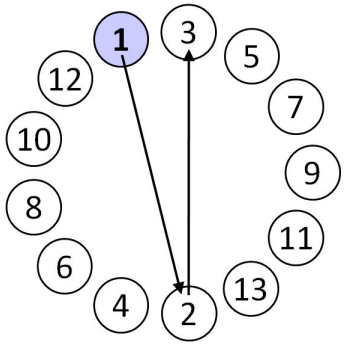


Figure 2.7: Two-dimensional Fitts' law task in ISO 9241-9.

$$ID = \log_2 \left(\frac{A}{W} + 1\right). \tag{2.1}$$

The main performance measure in ISO 9241-9 is throughput (TP, in bits/second or bps) which is calculated over a sequence of trials as the ID-MT ratio:

$$TP = \frac{ID_e}{MT}. \tag{2.2}$$

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:

$$ID_e = \log_2 \left(\frac{A_e}{W_e} + 1\right) \tag{2.3}$$

with

$$W_e = 4.133 \times SD_x. \tag{2.4}$$

The term SD_x is the standard deviation in the selection coordinates computed over a sequence of trials. For the two-dimensional task, selections are projected onto the

task axis, yielding a single normalized x-coordinate of selection 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 A_e instead of A is only necessary if there is an overall tendency for selections to overshoot or undershoot the target (see [20] for additional details).

Throughput is a potentially valuable measure of human performance because it embeds both the speed and accuracy of participant responses. Comparisons between studies are therefore possible, with the proviso that the studies use the same method in calculating throughput. Figure 2.8 is an expanded formula for throughput, illustrating the presence of speed and accuracy in the calculation.

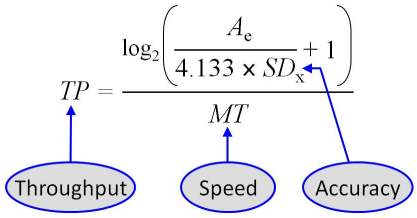


Figure 2.8: The calculation of throughput includes speed and accuracy.

Our testing used *GoFitts*[1], a Java application which incorporates *FittsTaskTwo* and implements the 2D Fitts' law task described above. *GoFitts* includes additional utilities such as *FittsTrace* which plots the cursor trace data captured during trials.

[1]http://www.yorku.ca/mack/GoFitts

2.6 Snake: Hands-free Edition

We have remodeled a the famous video game- Snake. We named our version of the game as *Snake: Hands-Free Edition*. The idea remains the same as the original game. In our version, a snake moves within the bounds of a surface. Two kinds of objects appear on this surface randomly. The white objects are *fruits*, the black objects are *poisonous* (see Figure 2.9). Colliding with a *fruit* will increase the snake's length and colliding with a *poisonous* object will kill the snake. The snake also dies if it collides with any wall or obstacle within the bounds of the surface.

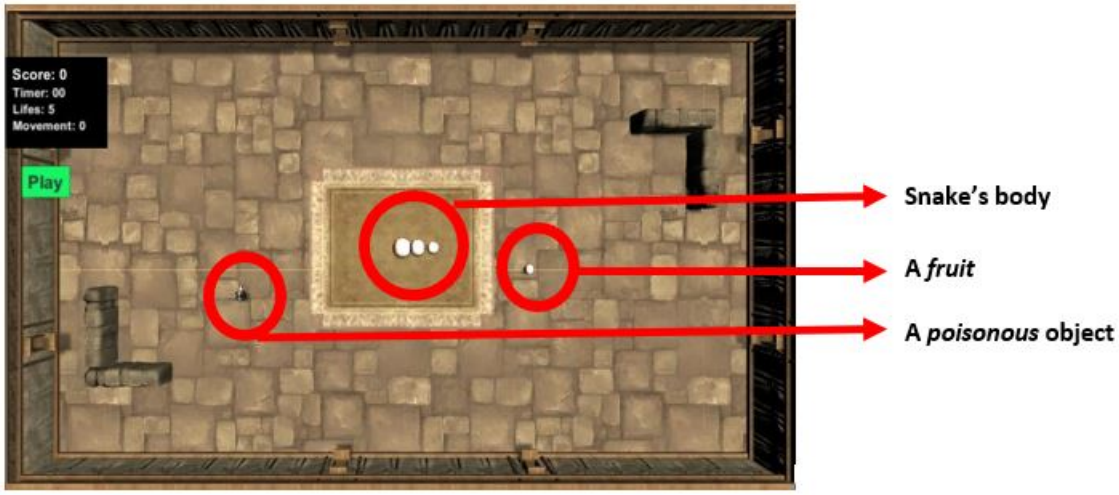


Figure 2.9: The snake, the *fruit* and the *poisonous* object of the game.

The snake's speed gradually increases in each trial. The maximum time limit for a trial is a minute. There are three input methods for the movement of the snake: keyboard, touchpad and *Camera Mouse*. While altering directions of the snake with a keyboard, a user needs to press the four arrow keys of a standard keyboard: up, down, right, and left. The game window is divided into four regions (i.e., left, right, up, down) to aid movement with the touchpad and *Camera Mouse* (see Figure 2.10).

These regions represent the functionality of the four arrow keys. A user needs to hover the cursor of the touchpad or *Camera Mouse* over these regions to change direction of the Snake using these regions.

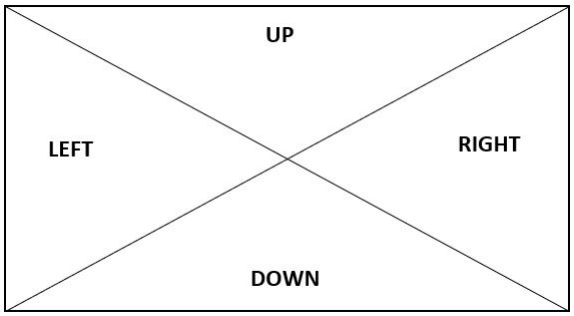


Figure 2.10: Four regions for cursor control.

Figure 2.10 shows how the game screen is divided into regions. If the cursor is in the area marked as left, the snake would move to the left. If the cursor is in the area marked as up, the snake would move to the up. If the cursor is in the area marked as right, the snake would move to the right. If the cursor is in the area marked as down, the snake would move to the down. One significant aspect of the snake's movement is, it is not allowed to move to an exact opposite direction from any direction it is moving towards. For example, if the snake is moving to the right, it cannot directly move to the left by pressing the keyboard's left arrow key or by hovering the cursor on the area marked as left with the touchpad or *Camera Mouse*. It would either have to go either up or down and then left to achieve movement in the left-direction. This idea is consistent with how the original *Snake* game is played.

Chapter 3

Literature Review

We now focus on previous research work on the three topics above, with interesting in the potential for accessible computing. The three topics are text entry, point-select tasks, and gaming. The literature review is organized by these three topics.

3.1 Research on Text-entry for Accessible Computing

Partridge et al. [23] introduced *TiltType*, a text-entry system for small devices. Users tilt the device to enter a character. One or more button presses are also required. The character chosen depends on the button pressed, and the direction and angle of the tilt. The size of the device is about the size of a wristwatch. A trade-off was discovered between the number of buttons and the number of tilt positions. They chose four buttons to accommodate all the English characters.

Widgor and Balakrishnan's [30] *TiltText* is another text-entry system that uses device tilt for user input. They compared *TiltText* with *Multitap* for mobile devices.

TiltText uses the standard 12-button mobile phone keypad with a low-cost tilt sensor. This system combines button presses and device tilt to determine which character is selected. Two different error types, button error and tilt error, were introduced. Button error is an error in aiming for a button. Tilt error is an erroneous tilt within the same button. *TiltText* had an overall error rate of 11% and *MultiTap* had an error rate of 3%. But the higher error rate of *TiltText* was mostly due to tilt error.

Wang et al. [29] present *TinyMotion*. This is a technique to detect movements of a cell-phone in real time by analyzing the image sequences captured by the built-in camera of the device. *TinyMotion* detects horizontal, vertical, rotational, and tilt movements. They re-created the accelerometer-based mobile input method developed by Widgor and Balakrishnan [30] and named it *Vision Tilt-Text*. This input method was found to be faster that *Multitap* within a few minutes of practice.

Cloud et al. [3] conducted an experiment with *Camera Mouse* that tested 11 participants, one with severe physical disabilities. The participants were tested on two applications, *EaglePaint* and *SpeechStaggered*. *EaglePaint* is a simple painting application that uses a mouse pointer. *SpeechStaggered* allows users to spell words and phrases by accessing five boxes that contain the English alphabet. Measurements for entry speed or accuracy were not reported; however, a group of participants wearing glasses showed better performance than a group not wearing glasses.

Text-entry research has focused on facial tracking in previous studies and the *Opti* keyboard layout has been tested against the *Qwerty* layout in numerous experiments as well. Our contribution is to present the first test of the *Opti* keyboard layout in a hands-free setup on a mobile platform (see Chapter 4 for further details).

3.2 Research on Point-select Tasks for Accessible Computing

Toyama [28] discusses a hands-free cursor control that uses real-time 3D face-tracking. In this system, users point their nose where they intend to position the cursor on a display. A framework called Incremental Focus of Attention (IFA) is used to robustly track facial position and orientation in real time. Significant findings include the observation that all users were able to hold the cursor on a 1×1 cm^2 box when the monitor was 50 cm from the users.

Corcoran et al. [4] discuss combining real-time face detection with eye-gaze tracking as a user input system. This technique requires a video feed from a low-resolution front-facing camera. Their apparatus did not use any wearable attachments or additional lighting. It was found that the slowest hardware configuration faced difficulties in achieving a frame rate of 30 fps. They acknowledge that some parts of their underlying work are present in commercial products as well.

Kaufman et al. [12] used an inexpensive eye movement-controlled user interface for 2D and 3D interaction. They used electro-oculography (EOG) instead of very expensive reflectance-based methods. Their system validated the viability of EOG in human-computer communication. They focused on creating a workable eye-controlled user interface comprising inexpensive, off-the-shelf components for detecting horizontal and vertical eye movement. The experiments were carried out on a 3×2 boxed menu. In most cases, the errors were inter-related: When an error occurred in horizontal detection it also accounted for a vertical error.

MacKenzie [18] evaluated eye tracking systems for computer input, noting that eye-tracking can emulate the functionality of a mouse. The evaluation followed ISO

9241-9, which lays out the requirements for non-keyboard input devices. Four selection methods were tested: dwell time, key selection, blink, and dwell-time selection. The throughput for dwell-time selection was 1.76 bits/s, which was 51% higher than the throughput for blink selection.

Gips et al. [7] developed *Camera Mouse* which used a camera to visually track any selected feature of the body such as the nose or tip of a finger. The tracking controls a mouse pointer on the computer screen. These were early days in the development of the system. At this stage, the system did not have any tracking history. Cloud et al. [3] describe an experiment with *Camera Mouse* with 11 participants, one with severe physical disabilities. Two application programs: *EaglePaint* and *SpeechStaggered* were used. *EaglePaint* is a simple painting application with the mouse pointer. *SpeechStaggered* is a program allows the user to spell out words and phrases from five boxes that contain the entire English alphabet. A group of subjects wearing glasses showed better performance in terms of elapsed time than a group of subjects not wearing glasses.

Betke et al. [1] describe further advancements with *Camera Mouse*. They examined various body features for robustness and user convenience. Twenty participants without physical disabilities were tested along with 12 participants with physical disabilities. Participants were tested for performance of *Camera Mouse* on two applications: *Aliens Game* which is an alien catching game requiring movement of the mouse pointer and *SpellingBoard*, a phrase typing application where typing was done by selecting characters with the mouse pointer. The participants not having any disabilities showed better performance with a normal mouse than *Camera Mouse*. Nine out of the 12 disabled participants showed eagerness in continuing to use the *Camera Mouse* system.

Magee et al. [22] discussed a multi-camera based mouse-replacement system.

They addressed the issue that an interface can lose track of a user's facial feature due to occlusion or spastic movements. Their multi-camera recording system can record synchronized images from multiple cameras. For the user study, a three-camera version of the system was used. Fifteen subjects were tested on a hands-free human-computer interaction experiment that included the work of Betke et al. [1] on *Camera Mouse*. They tracked a user's head movement with three simultaneous video streams and software called *ClickTester*. They report that users put more effort into moving the mouse cursor with their head when the pointer was in the outer regions of the screen.

Magee et al. [21] et al. did a user study with *Camera Mouse* where dwell-time click generation was compared with *ClickerAID*, which detects a single intentional muscle contraction with an attached sensor. An interactive evaluation tool called *FittsTaskTwo*[1] [19, p. 291] was used for testing the participants. Ten participants were tested with *ClickerAID* gaining a subjective preference over *Camera Mouse*.

Fitts' Law is a well-established protocol for evaluating target-selection tasks. And there have been Fitts' law experiments with facial tracking. But, HCI experiments can be unique in their design. This is our contribution in the second experiment. We chose a different design, modified the method to see how participants fare under unique conditions compared to existing studies (see Chapter 5 for further details).

[1]https://www.yorku.ca/mack/HCIbook/

3.3 Research on Gaming for Accessible Computing

Cuaresma and MacKenzie [5] compared two non-touch input methods for mobile gaming: tilt-input and facial tracking. They measured performance of 12 users with a mobile game named *StarJelly*. This is an endless runner-styled game. Players were tasked with avoiding obstacles and collecting stars in the game. A substantial difference was observed in the mean scores of the two input methods. Tilt-based input had a mean score of 665.8 and facial-tracking had a mean score of 95.1.

Roig-Maimó et al. [24] present *FaceMe*, a mobile head tracking interface for accessible computing. Participants were positioned in front of a propped-up *iPad Air*. Via the front-facing camera, a set of points in the region of the user's nose were tracked. The points were averaged, generating an overall head position which was mapped to a display coordinate. *FaceMe* is a picture-revealing puzzle game. A picture is covered with a set of tiles, hiding the picture. Tiles are turned over revealing the picture as the user moves her head and the tracked head position passes over tiles. Their user study included 12 able-bodied participants and four participants with multiple sclerosis. All able-bodied participants were able to fully reveal all pictures with all tile sizes. Two disabled participants had difficulty with the smallest tile size (44 pixels). *FaceMe* received a positive subjective rating overall, even on the issue of neck fatigue.

Roig-Maimó et al. [24] described a second user study using the same participants, interaction method, and device setup. Participants were asked to select icons on the *iPad Air*'s home screen. Icons of different sizes appeared in a grid pattern covering the screen. Selection involved dwelling on an icon for 1000 ms. All able-

bodied participants were able to select all icons. One disabled participant had trouble selecting the smallest icons (44 pixels); another disabled participant felt tired and was not able to finish the test with the 44 pixel and 76 pixel icon sizes.

UA-Chess is a universally accessible version of the Chess game developed by Grammenos et al. [8]. This game has four input methods: mouse, hierarchical scanning, keyboard, and speech recognition. The hierarchical scanning method is designed for users with hand-motor disabilities. This scanning technique has a special "marker" that indicates input focus. A user can shift focus using "switches" (e.g., keyboard, special hardware, voice control). After focusing on an object another "switch" is used for selection. It currently supports visual and auditory output. A key innovative feature of this game is that, it allows the multi-player functionality in an offline environment as well.

Computer gaming inputs range from simple interactions to very intense interactions. For our third user study, we tested a simple game in a hands-free setup. The focus was not on the complexity or visual sophistication of the game, but on how participants would fare in a recreational environment with *Camera Mouse*. Hence, we chose a simple game, Snake, just like the simple games evaluated by Cuaresma and MacKenzie [5] and Grammenos et al. [8]. See further details in Chapter 6.

Chapter 4

First Experiment: Comparing Hands-free and Hands-on Text Entry Methods for Mobile Devices

In our first experiment, we tested two keyboard layouts: Opti and Qwerty, over three input methods: touch, device tilt and head movement. This experiment was done on a Samsung S8+ Android device. An Android screen has elements and widgets which require either touch events or click events to function. The EFM software works only with click events. The acceleration value of EFM was set to 2 and the dwell time for selecting a character on a keyboard layout was set to 1 second (see Figure 4.1). For compatibility with EFM, keys in *Keyboard Tester* are implemented as Android buttons and are thus capable of generating click events when selected. We present the methodology, result analysis and discussions of this experiment below:

4.1 Methodology

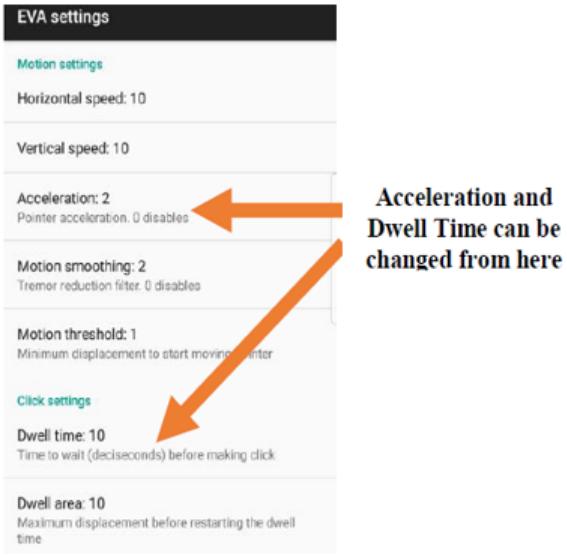


Figure 4.1: EFM settings.

4.1.1 Participants

We recruited 12 participants. Ten of the 12 participants were male, aged 23-27. Two were female, aged 19-23.

4.1.2 Apparatus

A Samsung *Galaxy S8+* Android device was used as the hardware. *Keyboard Tester* and EFM were used as software. *GoStats*[1] was used for the statistical analysis.

[1]https://tinyurl.com/y85xkvlm

4.1.3 Procedure

Keyboard Tester shows a choice of keyboard layouts (see Figure 4.2) at the beginning. Upon choosing the layout, the application screen shows a setup screen to choose testing parameters. Each participant was assigned parameters for layout and input method and entered three phrases for each of the three input methods: touch, head movement (i.e., hands-free) and device tilt. The phrases were selected randomly from a file. After completing the task on one layout, each participant moved on to the other layout. Touch is the conventional touch-based text-entry method (see Figure

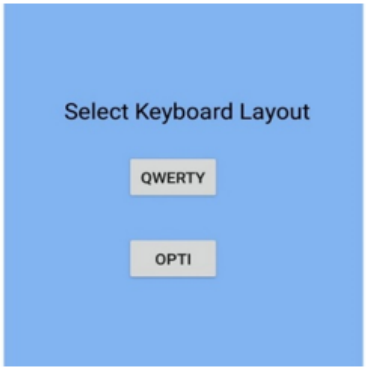


Figure 4.2: Keyboard Tester starting screen.

4.3-a). Head movement (i.e., hands-free) is for accessible computing where the user does not have to hold or move the device. The user moves their head in front of the device and enters phrases with the aid of EFM (see Figure 4.3-c). The device was supported on a laptop (see Figure 2.4). This affirms the head movement method as a hands-free method. But for device tilt, users kept their head steady and tilted the device to move the EFM cursor of EFM on the screen of *Keyboard Tester* (see Figure 4.3-b). After each phrase was entered, *Keyboard Tester* showed a popup with the entry speed, error rate, keystrokes per second (KSPC), the presented phrase and

the transcribed phrase (see Figure 4.3-d). Results were automatically collected and stored in .csv files after each phrase.

Figure 4.3: (a) Input method: touch, (b) Input method: device tilt, (c) Input method: head movement, (d) Screen-shot of *Keyboard Tester* application.

4.1.4 Design

This user study was a 2 × 3 within-subjects design. The independent variables and levels were as follows:

- Keyboard layout (*Opti, Qwerty*)

- Input method (touch, head movement, device tilt)

See Table 4.1 for details on the levels of input method. The independent variables were administered in counterbalanced order to offset learning effects. There were two groups for counterbalancing: layout group (beginning with *Opti* vs. beginning with *Qwerty,* six participants each) and method group (beginning with device tilt vs. beginning with head movement vs beginning with touch, four participants each). The dependent variables were error rate, entry speed (wpm), and keystrokes per character (KSPC). The total number of trials was 216 (12 participants $\times$ 2 keyboard layouts $\times$ 3 input methods $\times$ 3 phrases).

Table 4.1: Details of Input Methods.

Input Method	Input Source	EFM Dependency	Method Type
Touch	Touch sensor of Android device	Not dependent	Hands-on
Head movement	Front-facing camera of Android device	Dependent	Hands-free
Device tilt	Front-facing camera of Android device	Dependent	Hands-on

4.2 Results and Discussions: First Experiment

4.2.1 Error rate

The error rates are shown in Figure 4.4. The grand mean for error rate was 0.91%. The *Opti* layout produced a mean error rate of 0.98% compared to the mean error rate for *Qwerty* of 0.84%. Text-entry on *Qwerty* was about 17% less error-prone than text-entry on *Opti*. However, the effect of keyboard layout on error rate was not statistically significant ($F_{1,11} = 0.261$, ns).

By input method, the error rates were 0.89% (device tilt), 0.86% (head movement), and 0.97% (touch). The effect of input method on error rate was not statistically significant ($F_{2,22} = 0.032$, ns).

The effect of layout group on error rate was not statistically significant ($F_{1,10} = 1.061, p > .05$), nor was the effect of method group on error rate ($F_{2,9} = 2.08, p > .05$). Hence, counterbalancing achieved the desired results of offsetting learning effects.

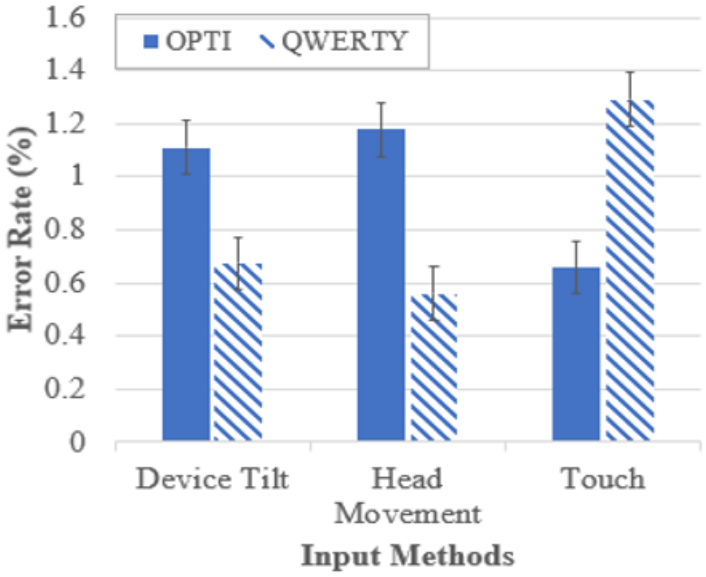


Figure 4.4: Error rate (%) by keyboard layout and input method. Error bars show ±1 *SE*.

Participants could correct their mistakes during text-entry. Both *Opti* and *Qwerty* had BACKSPACE buttons for correcting mistakes. Error rates were very low for both keyboard layouts and all three input methods. The third phrases were found to be more error-prone than the other two, as seen in Figure 4.5. The mean error rate of the third phrases for the device tilt method during text entry on *Opti* and the touch method during text entry on *Qwerty* was found to be much higher than

the initial two phrases.

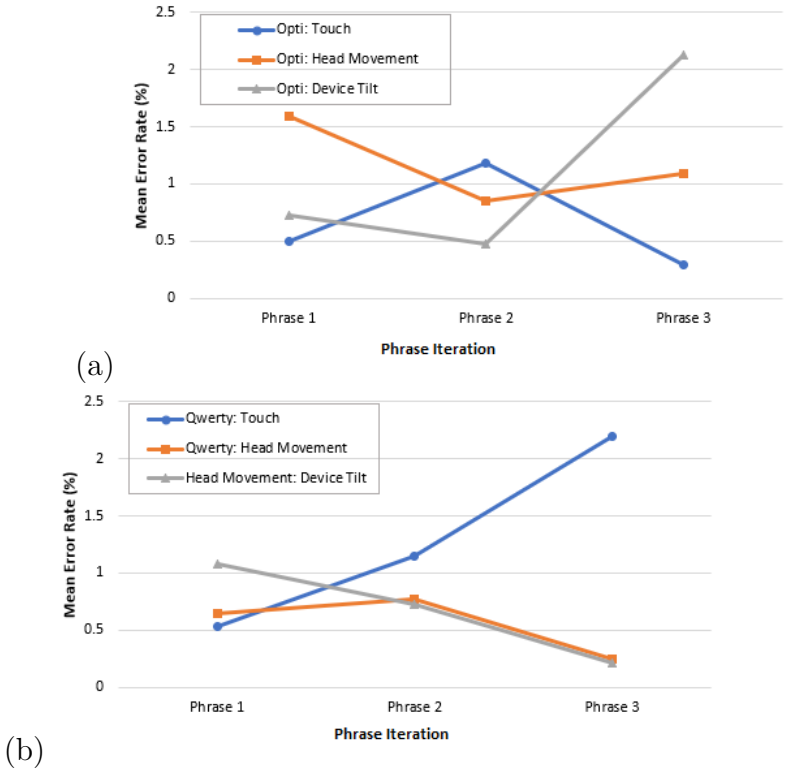


Figure 4.5: Mean error rate (%) of phrases by participants.

4.2.2 Entry Speed

The results for entry speed are seen in Figure 4.6. The grand mean for entry speed was 9.17 wpm. By keyboard layout, the means were 6.52 wpm (*Opti*) and 11.83 wpm (*Qwerty*). Clearly, *Opti* was slower than *Qwerty*. The effect of keyboard layout on entry speed was statistically significant ($F_{1,11} = 89.9, p < .0001$).

By input methods, the entry speeds were 4.66 wpm (device tilt), 4.30 wpm (head

movement), and 18.57 wpm (touch). The effect of input method on entry speed was statistically significant ($F_{2,22} = 106.1, p < .0001$).

As with error rate, the group effects were not significant for entry speed; hence, counterbalancing achieved the desired results. As seen in Figure 4.6, entry-speed

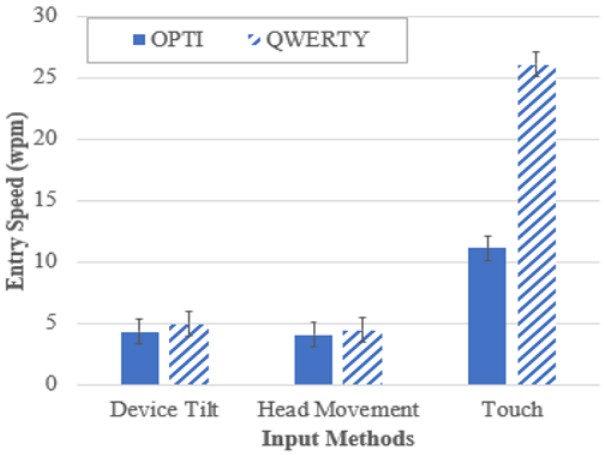


Figure 4.6: Entry speed (wpm) by keyboard layout and input method. Error bars show ± 1 *SE*.

is much higher for text-entry with the touch method on *Qwerty* (26.04 wpm) than any other combination of test parameters. This is due to multiple reasons. Firstly, participants are not as familiar with the *Opti* keyboard layout as they are with *Qwerty*. So, text-entry was naturally much slower on *Opti*. Secondly, text-entry with device tilt and head-movement (i.e., hands-free) are inherently slow input methods. Regardless of the keyboard layouts, the entry-speed slowed down substantially when participants were not touching the device screen.

Mean entry speeds for the phrases indicated that entry speeds for the touch method during text-entry on both keyboard layouts were higher for the first and third phrases (see Figure 4.7). The participants felt more comfortable with the

touch method and the excitement to start and finish an iteration of text-entry led to the higher mean entry speed values of the first and third phrases.

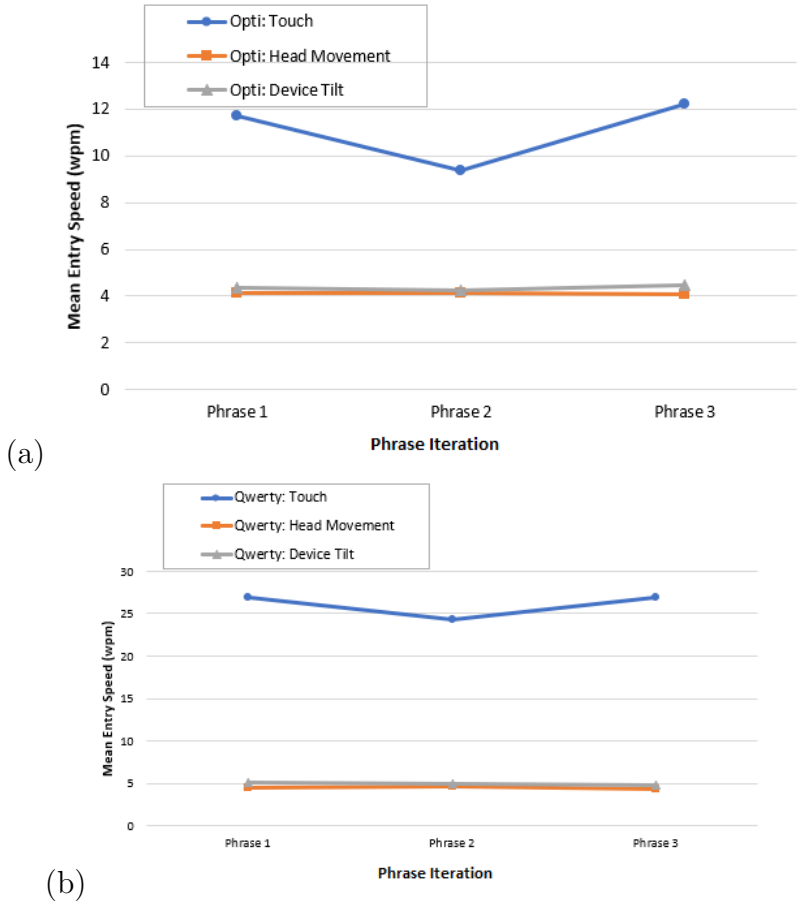


Figure 4.7: Mean entry speed (wpm) of phrases by participants.

4.2.3 Keystrokes Per Character (KSPC)

The results for KSPC are in Figure 4.8. The grand mean for KSPC was 1.10. KSPC for the *Opti* layout produced a mean of 1.07 compared to the mean KSPC of *Qwerty* layout at 1.13. *Opti* and *Qwerty* had very similar mean values of KSPC.

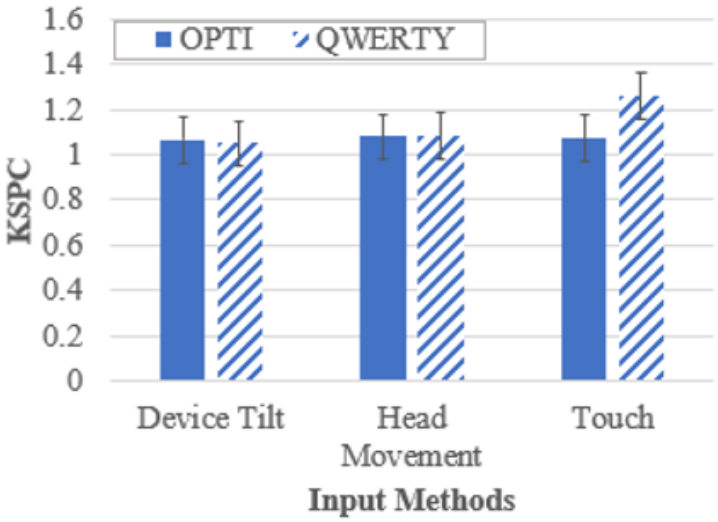


Figure 4.8: KSPC by keyboard layout and input method. Error bars show ±0.1 *SE*.

The device tilt input method produced a mean KSPC of 1.05. Head movement had a mean KSPC of 1.08. The touch input method had a mean KSPC of 1.16. Since the participants could correct their mistakes during text-entry, error rates were found to be very low. Hence, we used KSPC as a performance measure. Mean KSPC values were found to be close to 1 for both keyboard layouts and all three input methods. This indicates that participants did not make too many errors during the experiment. Thus, KSPC was not statistically analyzed. The mean KSPC for the three phrases doesn't maintain a particular trend, as seen in Figure 4.9.

4.2.4 Participant Feedback

Participant feedback was collected on a set of questionnaire items. They were asked about their preferred keyboard layout during the entire experiment and their preferred input method when not touching the device screen. They were also asked

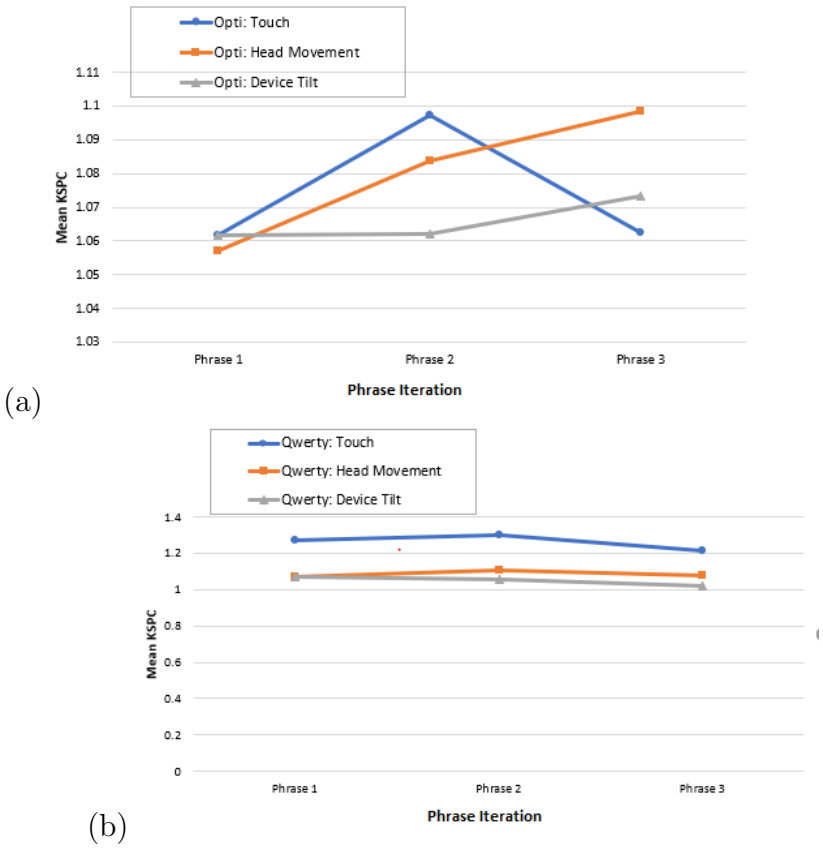


Figure 4.9: Mean KSPC of phrases by participants.

about their level of fatigue during the head movement (i.e., hands-free) phase of the experiment and about their view on the usefulness of this hands-free input system. Five-point Likert scale items were used to collect the feedback on fatigue and usefulness.

None of the 12 participants chose *Opti* as their preferred keyboard layout. Only two participants chose head movement as their preferred input method when not touching the device screen.

Chapter 5

Second Experiment: A Fitts' Law Evaluation of Hands-free and Hands-on Input on a Laptop Computer

The goal of our second user study was to empirically evaluate and compare two pointing methods (touchpad, *Camera Mouse*) in combination with two selection methods (tap, dwell). The hands-free method uses *Camera Mouse* with dwell-time selection. A 2D Fitts' law task was used with three movement amplitudes combined with three target widths. We present the methodology, result analysis and discussions of this experiment below:

5.1 Methodology

5.1.1 Participants

We recruited 12 participants. Nine were male aged 23-33 and three were female aged 23-29. All participants were from the local university community. None had prior experience using *Camera Mouse*.

5.1.2 Apparatus

An Asus *X541U* laptop was used as hardware. Both the built-in touchpad and the webcam provided input, depending on the pointing method. The touchpad was configured with the medium speed setting ("5") and with single-tap selection enabled.

The experiment tasks were presented using *GoFitts*, described earlier. The 2D task was used with 11 targets per sequence. Three amplitudes (100, 200, 400 pixels) were combined with three target widths (20, 40, 80 pixels) for a total of nine sequences per condition.

Selection was performed by the *GoFitts* software (not *Camera Mouse*). For dwell-time selection, a setting of 2000 ms was used. This somewhat long value was chosen after considerable pilot testing as it provided a balance between good selection and avoiding inadvertent selections.

Selection occurred after the cursor entered and remained in the target for 2000 ms. Errors were not possible. Visual feedback for the progress of the dwell timer was provided as a rotating arc inside the target. See Figure 5.1.

During dwell-time selection, if the cursor exited the target before the timeout, the timer was reset. When the cursor next entered the target, the software logged a "target re-entry" event.

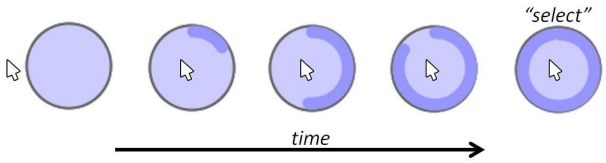


Figure 5.1: Visual feedback indicating the progress of the dwell timer.

For tap selection, participants were instructed to perform a single-tap with their finger on the touchpad surface.

5.1.3 Procedure

Participants were welcomed into the experiment. We explained the experiment to each participant and made them aware of the purpose of it. To make participants comfortable with the setup of the experiment and *Camera Mouse*, practice trials were allowed until they felt comfortable with the interaction.

Participants were instructed to select targets as quickly and accurately as possible, but at a comfortable pace. For each sequence, they were to proceed from the first to last target without hesitation. Between sequences, they could pause at their discretion. Figure 5.2 shows a participant doing the experiment task (a) using the touchpad with tap selection and (b) using *Camera Mouse* with dwell-time selection.

At the end of the experiment, participants provided feedback on a set of questions. They were asked about their preferred combination of pointing method and selection method. They also provided feedback on two 5-point Likert scale questions for physical fatigue and the overall rating of the hands-free phase.

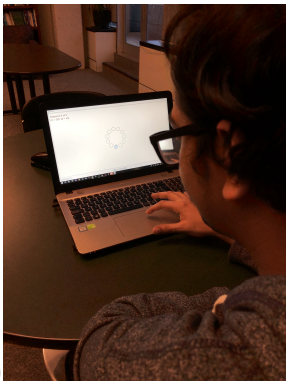
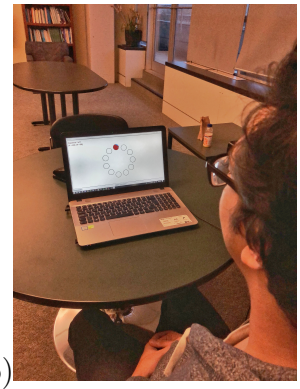
(a)  (b) 

Figure 5.2: Participant doing the experiment task (a) touchpad + tap selection (b) *Camera Mouse* + dwell-time selection.

5.1.4 Design

The experiment was a $2 \times 2 \times 3 \times 3$ within-subjects design. The independent variables and levels were as follows:

- Pointing method (touchpad, *Camera Mouse*)
- Selection method (tap, dwell)
- Amplitude (100, 200, 400 pixels)
- Width (20, 40, 80 pixels)

The primary independent variables were pointing method and selection method. Amplitude and width were included to ensure the conditions covered a range of task difficulties. The result is nine sequences for each test condition with *IDs* ranging from $\log_2(\frac{100}{80} + 1) = 1.17$ bits to $\log_2(\frac{400}{20} + 1) = 4.39$ bits. For each sequence, 11 targets appeared.

The dependent variables were throughput (bps), movement time (ms), error rate (%), and target re-entries (TRE, count/trial). There were two groups for counterbalancing, one starting with the touchpad and the other starting with *Camera Mouse*.

The total number of trials was 4752 ($= 2 \times 2 \times 3 \times 3 \times 11 \times 12$).

5.2 Results and Discussion: Second Experiment

Results are presented below organized by dependent variables. For all dependent variables, the group effect was not statistically significant ($p > .05$). This indicates that counterbalancing was effective in offsetting learning effects.

Cursor trace examples, Fitts' law regression models, and a distribution analysis of the selection coodinates are also presented. Statistical analyses were done using the *GoStats* application.[1]

5.2.1 Throughput

The grand mean for throughput was 1.22 bps. Pointing with the touchpad and *Camera Mouse* had mean throughputs of 1.70 bps and 0.75 bps, respectively. The effect of pointing method on throughput was statistically significant ($F_{1,10} = 117.8, p < .0001$). Clearly, doing the experiment task with *Camera Mouse* was more difficult than with the touchpad. Of course, there is no expectation that hands-free point-select interaction would compete with hands-on point-select interaction.

During pointing with the touchpad, selecting with tap and dwell had mean throughputs of 2.30 bps and 1.10 bps, respectively. While pointing with *Camera Mouse*, selecting with tap and dwell had mean throughputs of 0.85 bps and 0.65

[1] http://www.yorku.ca/mack/GoStats

bps, respectively. See Figure 5.3. The effect of selection method on throughput was statistically significant ($F_{1,10} = 93.0, p < .0001$).

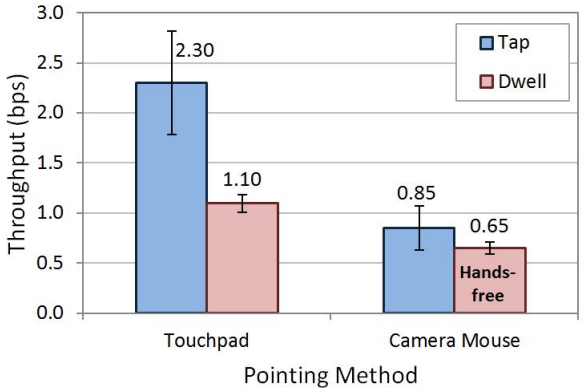


Figure 5.3: Throughput (bps) by selection method and pointing method. Error bars show ±1 *SD*.

The lowest throughput of 0.65 bps was for the *Camera Mouse* with dwell-time selection – hands-free interaction. This value is low, but is expected given the pointing and selection methods employed.

5.2.2 Movement Time and Error Rate

Since throughput is a composite measure combining speed and accuracy, the individual results for movement time and error rate are less important. They are briefly summarized below. See Figure 5.4.

The grand mean for movement time was 3026 ms per trial. See Figure 5.4a. The effects on movement time were statistically significant both for pointing method ($F_{1,10} = 395.1, p < .0001$) and for selection method ($F_{1,10} = 93.0, p < .0001$). Note in Figure 5.4a the long movement time of 5012 ms for *Camera Mouse* with dwell-time selection. As errors were not possible with dwell-time selection, the long movement

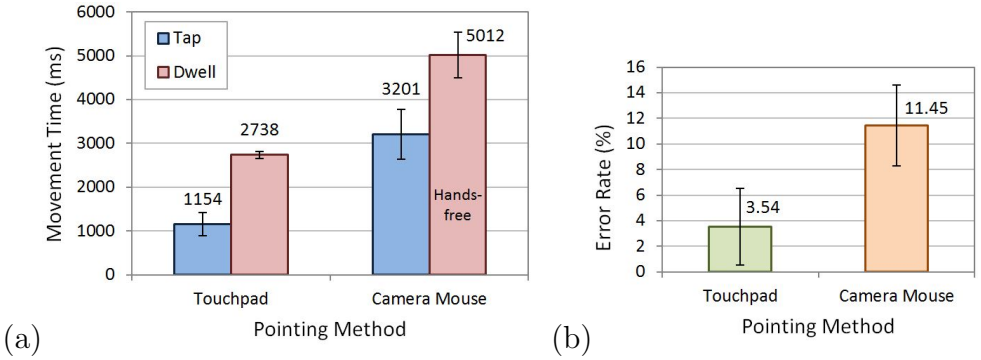


Figure 5.4: Results for speed and accuracy (a) movement time by pointing method and selection method (b) error rate by pointing method with tap selection.

time is likely caused by participants having difficulty maintaining the cursor inside the target for the required dwell-time (2000 ms). This point is examined in further detail below in the analyses for target re-entries.

The grand mean for error rate was 3.75%. See Figure 5.4b. The figure only shows the results by pointing method using tap selection, since errors were not possible for dwell-time selection. The effect of pointing method on error rate was statistically significant ($F_{1,10} = 67.3, p < .0001$).

5.2.3 Target Re-entries (TRE)

The grand mean for target re-entries (TRE) was 0.21 re-entries per trial. The implication is that for approximately one in every five trials the cursor entered the target, then left and re-entered the target. Sometimes this occurred more than once per trial.

Pointing with the touchpad and *Camera Mouse* had mean TREs of 0.09 and 0.31, respectively. So, TRE was about 3× higher for *Camera Mouse*. The effect of

pointing method on TRE was statistically significant ($F_{1,10} = 10.2, p < .01$).

During pointing with the touchpad, selecting with tap and dwell had mean TREs of 0.08 and 0.11, respectively. While pointing with *Camera Mouse*, selecting with tap and dwell had mean TRE of 0.18 and 0.46, respectively. See Figure 5.5. The effect of selection method on TRE was statistically significant ($F_{1,10} = 27.7, p < .0005$). Further discussion on target re-entries continues below in an examination of the trace paths for the cursor during pointing.

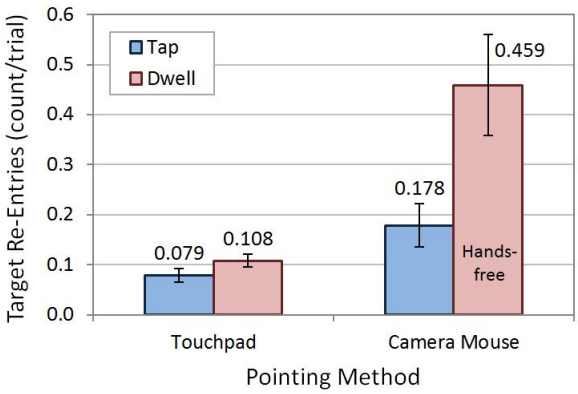


Figure 5.5: Target re-entries (count/trial) by selection method and pointing method. Error bars show ±1 *SE*.

5.2.4 Cursor Trace Examples

The high value for TRE with *Camera Mouse* warrants further investigation. This was done by examining the cursor trace files generated by *GoFitts*. Cursor movements were relatively clean for the touchpad pointing method. It was a different story for *Camera Mouse*, however, where some erratic cursor movement patterns were observed. For comparison, Figure 5.6 provides two examples. Both are for dwell-time selection with $A = 200$ pixels and $W = 20$ pixels. Figure 5.6a is for pointing

with the touchpad, while Figure 5.6b is for pointing with *Camera Mouse*.

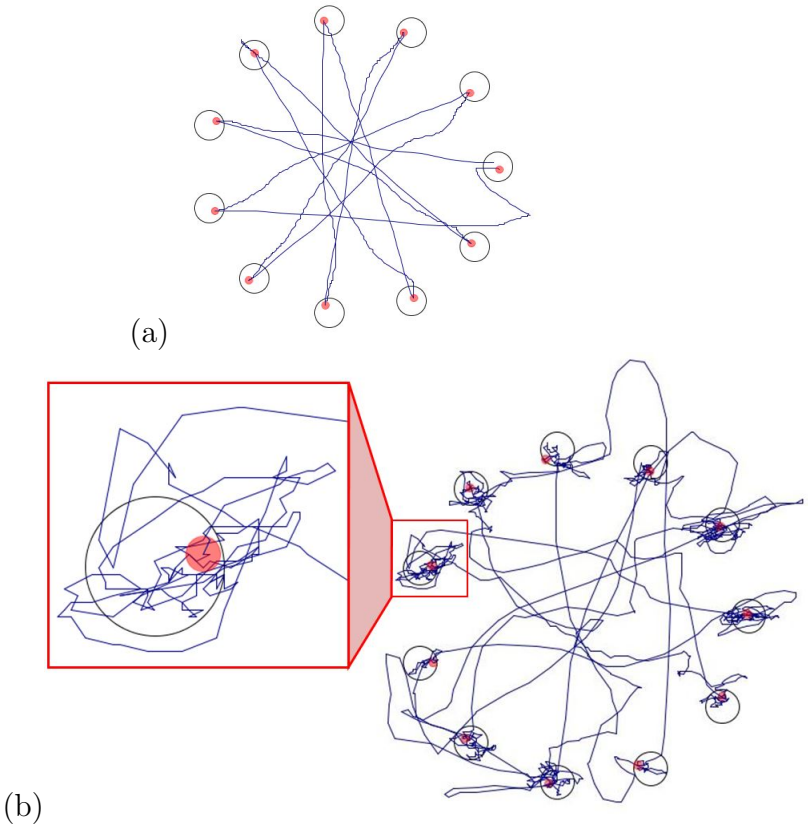


Figure 5.6: Cursor trace examples for dwell-time selection with $A = 200$ pixels and $W = 20$ pixels. The pointing methods are (a) touchpad and (b) *Camera Mouse*. See text for discussion.

It is evident that the cursor movement paths were more direct for the touchpad (Figure 5.6a) than for *Camera Mouse* (Figure 5.6b). In fact, the difference is dramatic, as seen in the call-out in Figure 5.6b. This particular trial had $MT = 14199$ ms with six target re-entries. Clearly, the participant had considerable difficulty keeping the cursor inside the target for the 2000-ms interval required for dwell-time

selection.

5.2.5 Fitts' Law Models

To test for conformance to Fitts' law, we built least-squares prediction equations for each test condition. The general form is

$$MT = a + b \times ID \tag{5.1}$$

with intercept a and slope b. See Table 5.1. The most notable observation in the table is the very high intercepts for the dwell models. Ideally, intercepts are 0 (or ≈ 0) indicating zero time to complete a task of zero difficulty, which has intuitive appeal. However, large intercepts occasionally occur in the literature. A notable case is the intercept of 1030 ms in Card et al.'s Fitts' law model for the mouse [2, p. 611].

Table 5.1: Fitts' law models

Condition	Intercept, a (ms)	Slope, b (ms/bit)	Correlation (r)
Touchpad + tap	699.2	171.4	.9124
Touchpad + dwell	2270.5	176.6	.9653
Camera Mouse + tap	404.5	1055.1	.9622
Camera Mouse + dwell	1135.9	1462.7	.8627

Scatter plot and regression line examples are seen in Figure 5.7 for pointing using *Camera Mouse*. Although the tap model provides a good fit ($r = .9622$, Figure 5.7a), the dwell-time model is a much weaker fit ($r = .8627$, Figure 5.7b). Behaviour was clearly more erratic in the *Camera Mouse* + dwell condition.

Table 5.2: Lilliefors normality test on selection coordinates by trial sequence

Condition	Sequences	Normality Hypothesis	
		Rejected	Not-rejected
Touchpad + tap	108	25	83
Touchpad + dwell	108	3	105
Camera Mouse + tap	108	15	93
Camera Mouse + dwell	108	7	101
Total	432	50	382

5.2.6 Distribution of Selection Coordinates

The calculation of throughput uses the effective target width (W_e) which is computed from the standard deviation in the selection coordinates for a sequence of trials (see Eq. 2.4). There is an assumption that the selection coordinates are normally distributed. To test the assumption, we ran normality tests on the x-selection values, as transformed onto the task axis. A test was done for each sequence of trials. We used the Lilliefors test available in *GoStats*. The results are seen in Table 5.2.

As seen in Table 5.2, the user study included 432 sequences of trials (12 participants × 2 pointing methods × 2 selection methods × 3 amplitudes × 3 widths). Of these, 382, or 88.4%, had selection coordinates deemed normally distributed. Thus, the assumption of normality is generally held. The best results in Table 5.2 are for dwell-time selection; however, this is expected since all the selection coordinates were inside the targets. For some reason, the touchpad with tap selection had 25 of 108 sequences (23.1%) with selection coordinates considered not normally distributed.

5.2.7 Participant Feedback

Participants were asked to provide feedback on the experiment and indicate their preferred test condition. Eight out of 12 participants chose the touchpad with tap selection as their preferred test condition. *Camera Mouse* with tap selection was preferred by two of the 12 participants. *Camera Mouse* with dwell-time selection and touchpad with dwell-time selection were preferred by one participant each.

Participants also provided responses to two 5-point Likert scale questions. One question was on the participant's level of fatigue with *Camera Mouse* (1 = *very low*, 5 = *very high*). The mean response was 2.4, closest to the *low* score. The second question was on the participant's rating of the hands-free phase of the experiment (1 = *very poor*, 5 = *very good*). The mean response was 3.4, just slightly above the *normal* score. So, interaction with *Camera Mouse* fared reasonably well, but there is clearly room for improvement.

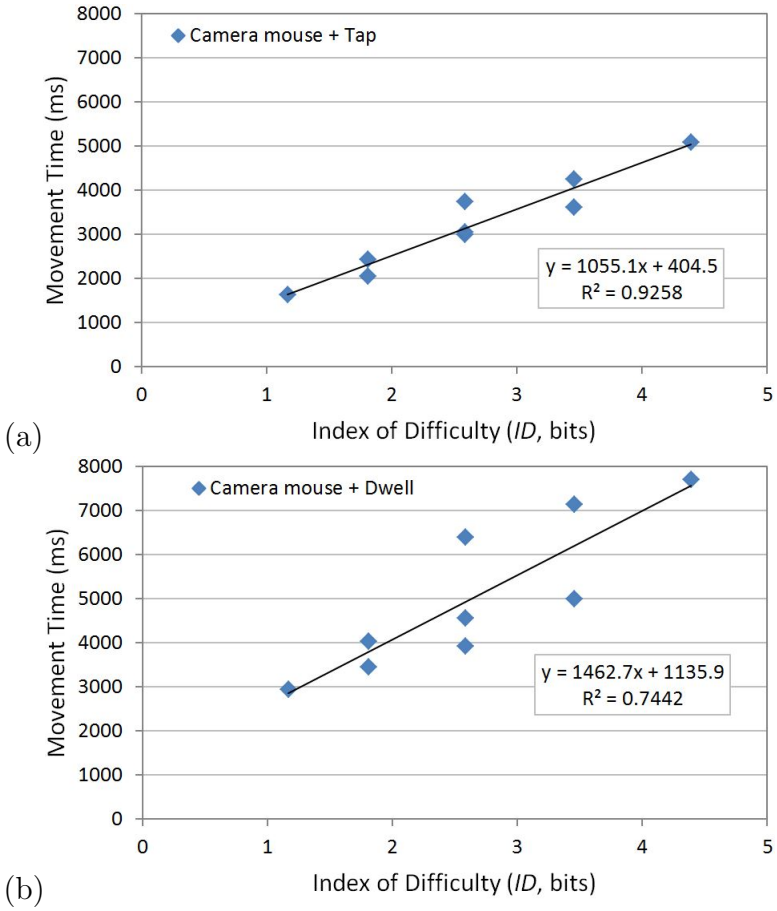


Figure 5.7: Example Fitts' law models for *Camera Mouse*. Selection using (a) tap or (b) dwell.

Chapter 6

Third Experiment: Evaluating Hands-on and Hands-free Input Methods for a Simple Game

Our third experiment was based on a game: *Snake Hands-free Edition*. We re-created a version of this famous game. It was played on three different input methods: keyboard, touchpad, and *Camera Mouse*. The first two methods are hands-on and the last one is hands-free. We present the methodology, result analysis and discussions of this experiment below:

6.1 Methodology

6.1.1 Participants

We recruited 12 participants for this experiment. Eight were male and four were female. The participants were university students at undergraduate and graduate

levels. They belonged to different demographic regions such as Bangladesh, Canada, India, and Pakistan.

6.1.2 Apparatus

For hardware, we used an *ASUS X541U series* laptop which has a built-in web-cam. The experiment software was based on our *Snake Hands-free Edition* game, as described earlier (see section 2.6). We used *Camera Mouse* for facial tracking. Two software tools – *GoFitts*[1] and *GoStats*[2] – were used for statistical analysis. We used the *Processing*[3] tool to generate trace files of the snake's movement and corresponding cursor movement for the touchpad and *Camera Mouse*.

6.1.3 Procedure

The experimenter explained the steps of the experiment to each participant and demonstrated with a few practice trials. The counterbalancing group for each participant was chosen at random. Each participant was allowed some practice trials which were not part of the result analysis. Participants were tested over three sessions, one for each input method (see Figure 6.1 parts a, b, and c). A maximum of two sessions were allowed within a day as long as there was a break of at least two hours between the sessions. Two consecutive sessions were not separated by more than two days. Each trial was one minute in duration.

[1]https://www.yorku.ca/mack/FittsLawSoftware/doc/index.html?GoFitts.html
[2]https://www.yorku.ca/mack/GoStats/
[3]https://processing.org/

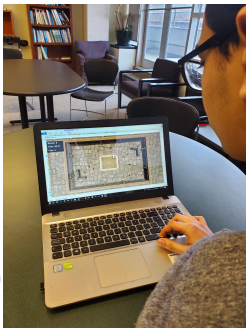
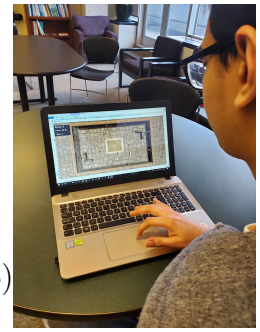
(a)  (b)  (c) 

Figure 6.1: A participant taking part in the Snake game with (a) keyboard, (b) touchpad, and (c) *Camera Mouse*.

6.1.4 Design

The user study was a 3×8 within-subjects design. The independent variables and levels were as follows:

- Input method (keyboard, touchpad, *Camera Mouse*)
- Number of blocks (1, 2, 3, 4, 5, 6, 7, 8)

There were three dependent variables: score, completion time (in seconds), and number of movements (count). The total number of trials was 1440 ($3 \times 8 \times 5 \times 12$). The three input methods were counterbalanced with four participants in each group to offset learning effects.

6.2 Results and Discussions

6.2.1 Score

Whenever the snake eats a fruit, the score increases by one. The mean score for the keyboard method was 5.89, the mean score for the touchpad was 3.34, and *Camera*

Mouse had the lowest mean score at 1.94 (see Figure 6.2).

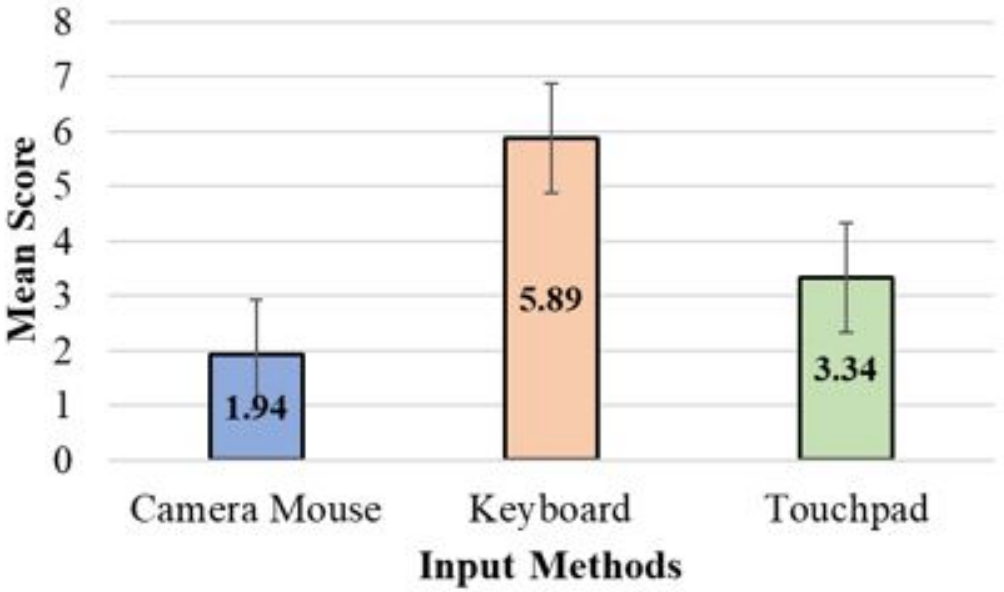


Figure 6.2: Mean score by input methods. Error bars indicate $\pm 1SE$.

The effect of group on score was statistically significant ($F_{2,9} = 4.619, p < .05$). Counterbalancing did not achieve the desired results in this case. The effect of input method on score was statistically significant ($F_{2,18} = 83.569, p < .0001$). The effect of block on score was not statistically significant ($F_{7,63} = 1.225, p > .05$).

As seen in Figure 6.2, the mean score is much higher for the keyboard compared with the touchpad and *Camera Mouse* method. The keyboard method is the closest simulation of the original snake game on Nokia mobile phones, where users had to press physical buttons to play the original game. This similarity likely played a part in the keyboard being the best among input methods in terms of score. The low scores with *Camera Mouse* are likely due to the newness of the method while playing such a game.

6.2.2 Completion Time

Completion time signifies the time in seconds for which the snake was alive in each trial. In each trial, after 60 seconds, the snake would die automatically and the user would be proceed to the next trial. The mean completion time for the keyboard method was 40.79 seconds, the mean completion time for the touchpad method was 39.32 seconds, and *Camera Mouse* had a mean completion time of 35.38 seconds (see Figure 6.3).

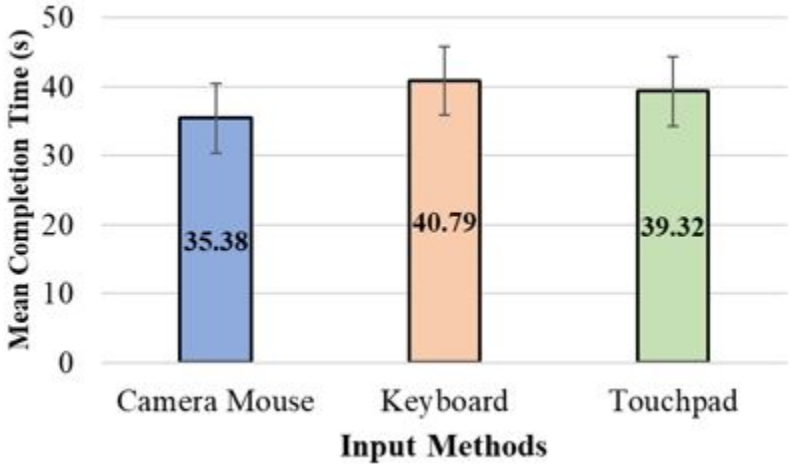


Figure 6.3: Mean completion time (s) by input methods. Error bars indicate $\pm 5SE$.

The effect of group on completion time was statistically significant ($F_{2,9} = 8.059, p < .01$). Counterbalancing did not achieve the desired results in this case. However, the effect of input method on completion time was not statistically significant ($F_{2,18} = 2.185, p > .05$). The effect of block on completion time was not statistically significant ($F_{7,63} = 0.865, ns$).

As stated earlier, the completion time for each trial signifies the in-game lifespan of the snake. The snake could die by hitting a poisonous object (black) or hitting a

wall unless the allotted 60 seconds for a trial are over. The black objects appear on the game-screen randomly. Their appearance does not have any periodic or positional consistency and thus brings an element of surprise to the player. This is true for all three input methods, hence as seen in section 6.3, the mean completion times for each input method are not that different from each other.

6.2.3 Number of Movements

The number of movements were tallied as counts. Each time the snake changed direction, the number of movements increased by one. The mean number of movements for the keyboard method was 73.90, the mean number of movements for the touchpad method was 28.18, and *Camera Mouse* had a mean number of movements of 21.57 (see Figure 6.4).

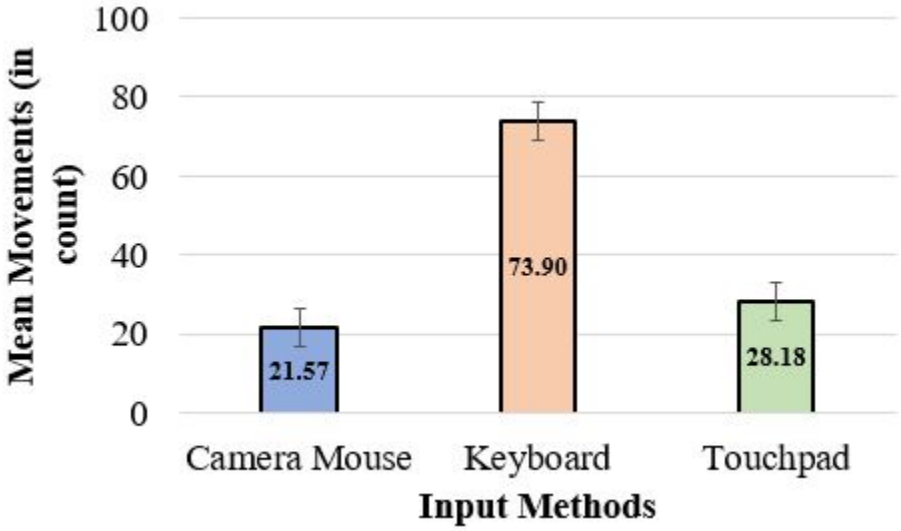


Figure 6.4: Mean completion time (s) by input methods. Error bars indicate $\pm 5SE$.

The effect of group on movements was not statistically significant ($F_{2,9} = 2.101, p >$

.05, whereas the effect of input method on movements was statistically significant ($F_{2,18} = 65.157, p < .0001$). The effect of block on movements was not statistically significant ($F_{7,63} = 0.669, ns$).

Keyboard was the best performing input method for number of movements as well. The participants discovered that tapping diagonally positioned keys (i.e., left-up, up-right, left-down, down-right etc) results in swift movement of the snake. They used this to move the snake faster in moving toward *fruits*; this resulted in a high number of movements with the keyboard. Participants struggled to bring this swift movement into effect with the touchpad or *Camera Mouse*, as covering the directional regions (see Figure 2.10) was obviously not as easy as changing directions with the keyboard.

6.2.4 Learning

While analyzing learning over the eight blocks of testing, we found an improvement in some cases, but not all cases.

Beginning with the keyboard score (see Figure 6.5a), there was very little learning effect for score during the first three blocks and the final two blocks. Otherwise, there were dips in the score. The reason behind this is a combination of familiarity with the game and fatigue coming into effect after four blocks. It is significant to note that many participants took a break of five minutes after the first four blocks of each session. Hence, when they restarted they did better but when fatigue increased, their performance took a dip again. Figure 6.5b shows that completion time gradually decreased across the eight blocks with the keyboard. The fatigue effect is evident here. It also signifies, looking back at Figure 6.5a, that participants achieved a better score with less completion time during the final 2-3 blocks of the experiment while

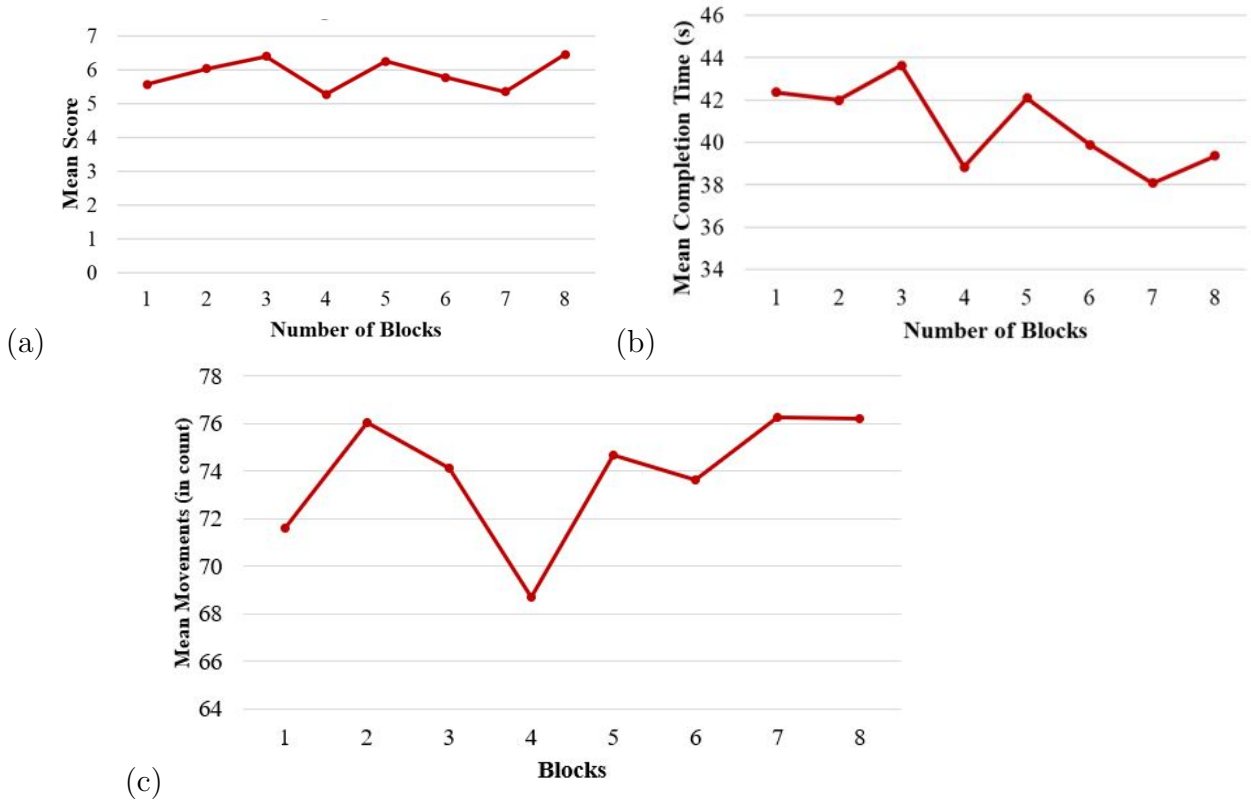


Figure 6.5: Learning over eight blocks with the keyboard for (a) score, (b) completion time (s), and (c) movements (in counts).

using the keyboard. For the number of movements with the keyboard, we see some learning taking place in Figure 6.5c during the final 2-3 blocks but there is a large dip during the blocks in the middle of the experiment.

For score using the touchpad, the learning effects line was mostly flat. But very little learning was observed for completion time (see Figure 6.6a) and number of movements (see Figure 6.6b) while using the touchpad.

With *Camera Mouse*, there was no significant improvement with practice for

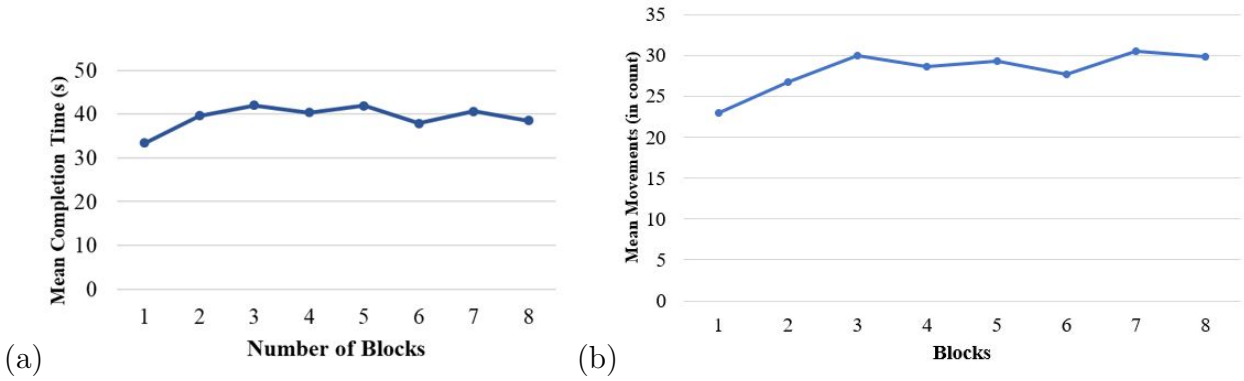


Figure 6.6: Learning over eight blocks with touchpad for (a) completion time (s), and (b) movements (in counts).

score. But completion time gradually decreased over the first few blocks and it fluctuated across the rest of the blocks (see Figure 6.7a). Similar remarks can be made about movements as well while using the *Camera Mouse* (see Figure 6.7b).

6.2.5 Traces of the Snake and Cursor

To further support our idea of dividing the game-screen into regions (see Figure 2.10 for touchpad and *Camera Mouse* control), we generated trace files. For the simplicity of presentation, we show some trace file examples in Figure 6.8 and Figure 6.9. This trace file was generated from a relatively short trial from one of the *Camera Mouse* sessions. The red and and blue dots in Figure 6.8 and Figure 6.9 decreased in size as time progressed along the trial. Note that the snake followed the cursor as the final point in the cursor's trace file was in the *UP* region and the final point of the snake's trace file shows that the snake was indeed going upwards. An overlapping trace file for both the cursor and the snake is depicted in Figure 6.9. A similar image for a

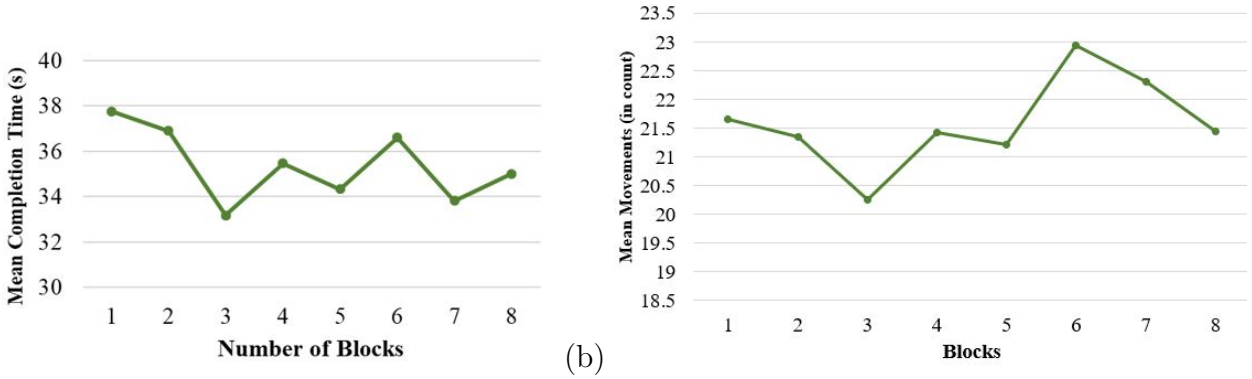


Figure 6.7: Learning over eight blocks with *Camera Mouse* for (a) completion time (s), and (b) movements (in counts).

longer lasting trial is shown in Figure 6.11.

6.3 Participant Feedback

We collected participant feedback on a set of questionnaires. Five out of the 12 participants had no prior experience of using *Camera Mouse*. When asked about their preferred method of input, 11 out of 12 participants chose the keyboard; only one participant chose touchpad.

Participants also provided responses on two 5-point Likert scale questions. One question was on the participant's level of fatigue with *Camera Mouse* (1 = *very low*, 5 = *very high*). The mean response was 2.67, closest to the *moderate fatigue* score. The second question was on the participant's rating of the hands-free phase of the experiment (1 = *very poor*, 5 = *very good*). The mean response was 3.25, just slightly above the *normal* score. Hence, it can be noted that the interaction with *Camera Mouse* fared well.

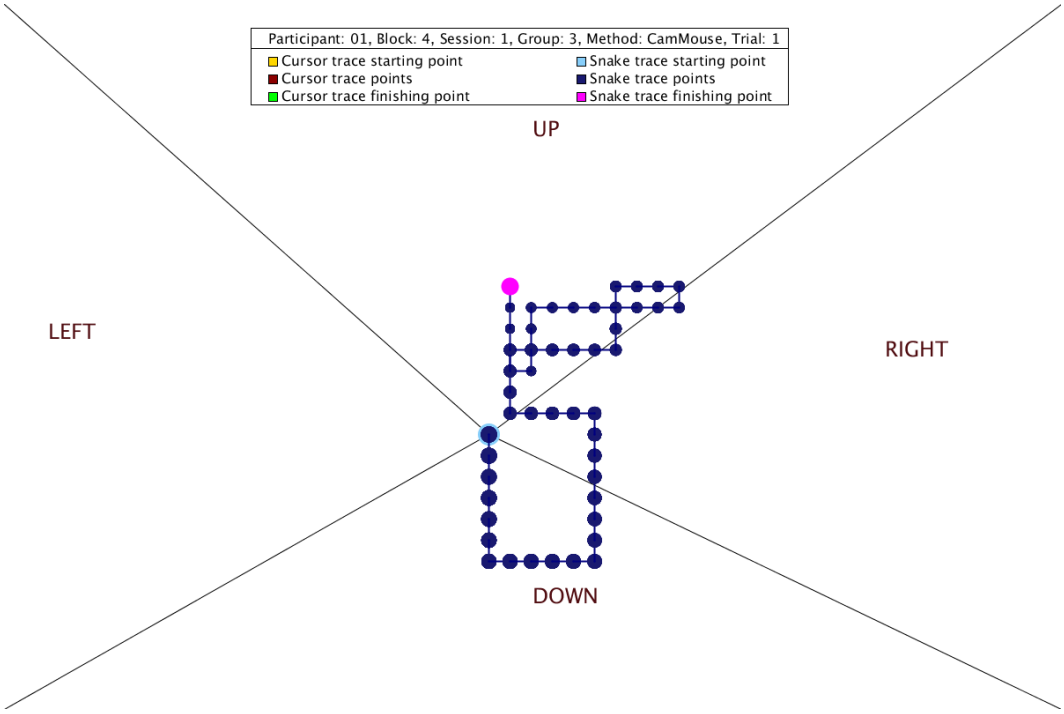


Figure 6.8: Trace file for the snake's movement.

We also asked the participants if they could name any other games that they think can be played with the *Camera Mouse*. Some interesting answers were given. Participants suggested games such as Temple Rush, Subway Surfer, and Point Of View Driving as some possible candidates. However, five of the 12 participants thought *Camera Mouse* could not be used in any other games. Participants were also asked about aspects of *Camera Mouse* that they struggled with. The responses were keeping track of the cursor, figuring out the required degree of head movement, horizontal movement, vertical movement, sensitivity of the cursor, etc. 'Keeping track of the cursor' received five responses, 'figuring out the required degree of head movement' received fives responses as well. 'Sensitivity of the cursor' received nine responses.

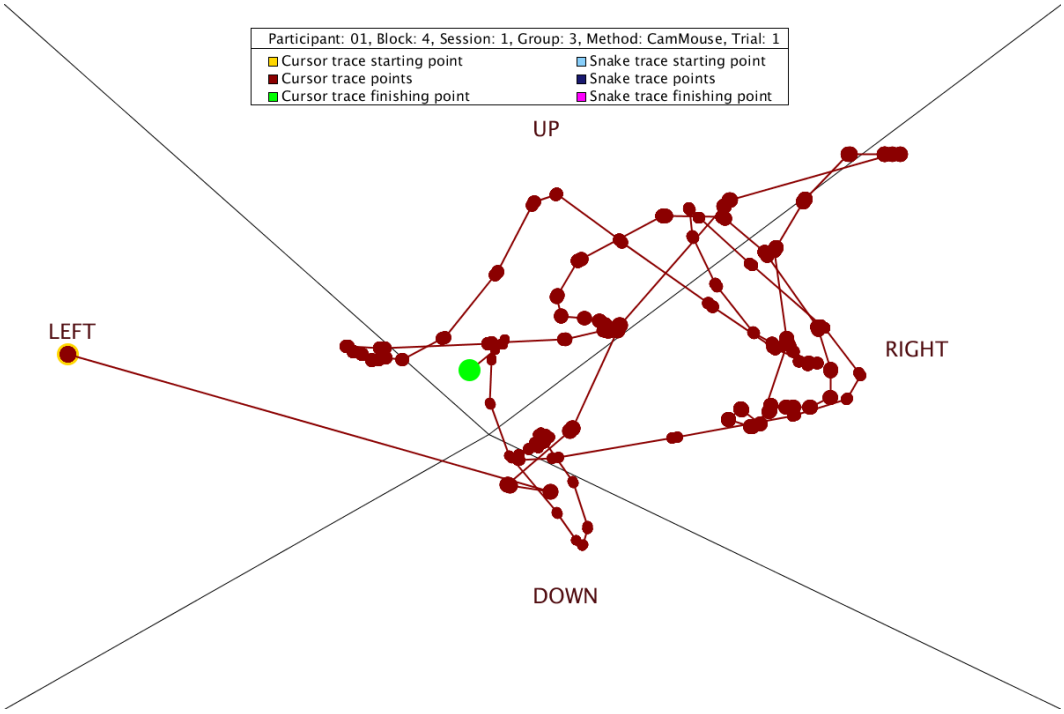


Figure 6.9: Trace file for cursor movement.

6.4 Case Study

We conducted a case study along with our user study with a participant who has the physical condition: mild cerebral palsy. The participant is an undergraduate student. He is able to walk and has some use of his hands, but he does not have as much control in his hands as the 12 participants of the user study. Hence, he was only asked to do the hands-free session of the experiment (see Figure 6.12). He took part in all eight blocks of the hands-free session. The hardware, software, and procedure also remained same as the user study for this case study.

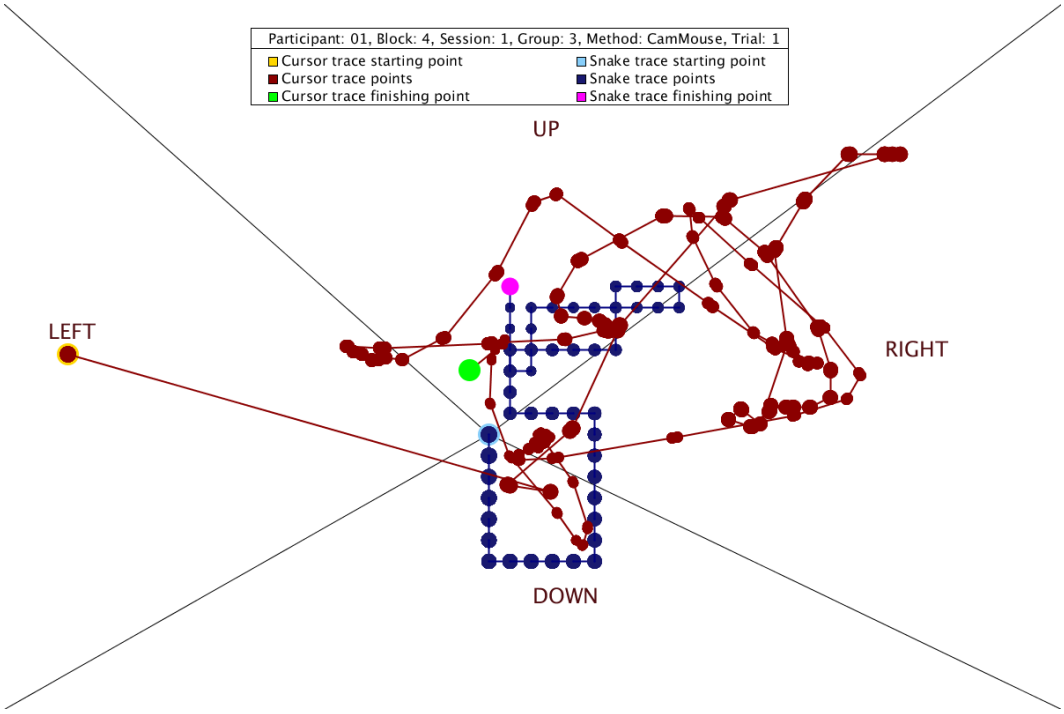


Figure 6.10: Trace file for the snake's movement and cursor movement.

6.4.1 Design

This was a 1×8 single-subject design. We had two factors for the case study. Their names and levels are:

- Input method (*Camera Mouse*)
- Blocks (1, 2, 3, 4, 5, 6, 7, 8)

Of course, input method is not a factor, per se, since only one level was used. It is reported as such to be consistent with the earlier user study.

The performances measures were once again score, completion time (s), and number of movements (in counts).

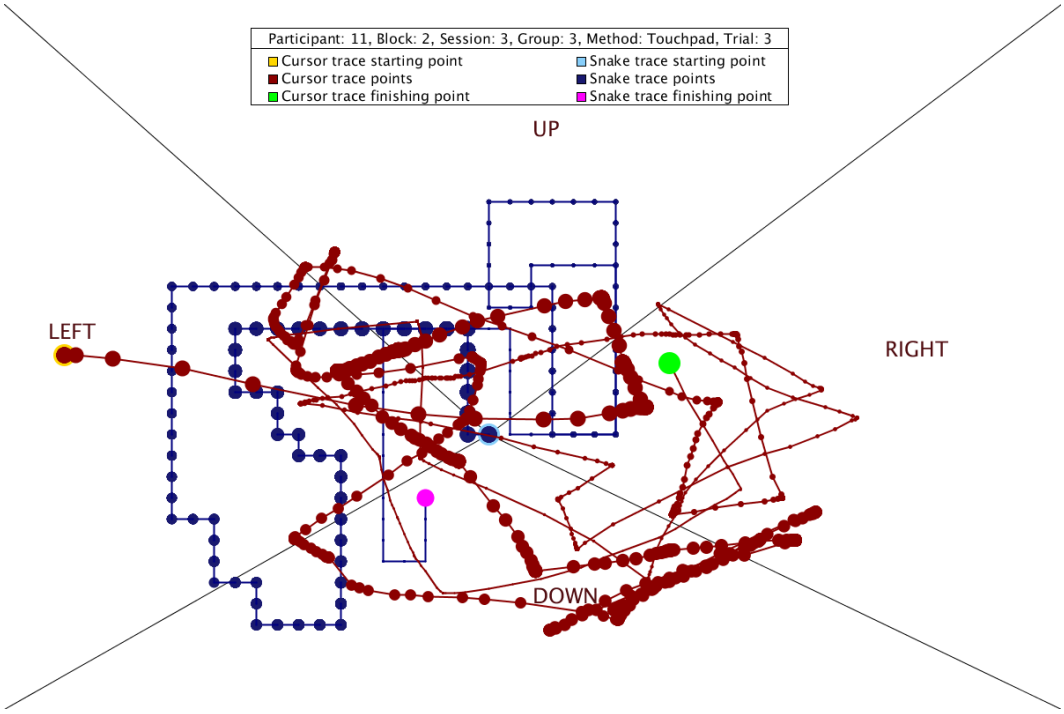


Figure 6.11: Trace file for the snake's movement and cursor movement (longer lasting trial).

6.4.2 Results and Discussion

We compared the performance measures of the hands-free user study with the performance measures of the case study. As seen in Figure 6.13-(a), the physically-challenged participant had a much lower mean score of 0.55 compared to the mean score for the hands-free session in the user study, which was 1.94. In terms of completion time (s), the case study had a mean completion time (s) of 15.5 seconds whereas the user study's mean completion time for the hands free session was 21.57 seconds (see Figure 6.13-(b)). The case study had mean number of movements at 21.08 but the user study had mean number of movements at 35.38 (see Figure 6.13-(c)). The

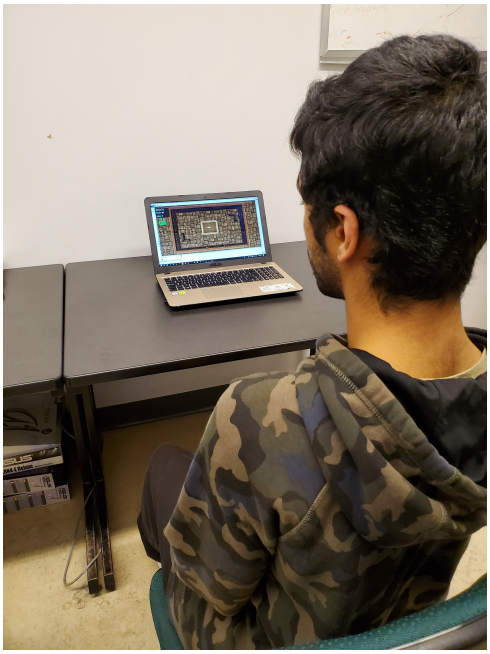


Figure 6.12: Case study participant taking part in the experiment.

physical condition of the participant of the case study, obviously did not allow him to be as flexible as the regular participants of the user study, thus lower performance mesure mean values were observed during the case study.

The participant displayed very little learning in terms of completion time (see Figure 6.14-(b)), but his performance fluctuated a lot in terms of score (see Figure 6.14-(a)). There was no significant learning effect observed for number of movements during the case study.

6.4.3 Case study Participant Feedback

The participant had no prior experience of using the *Camera Mouse* and did not mention any other game which can be played with *Camera Mouse*. He affirmed he did not feel much fatigue during the session. When was asked to rate his *Camera*

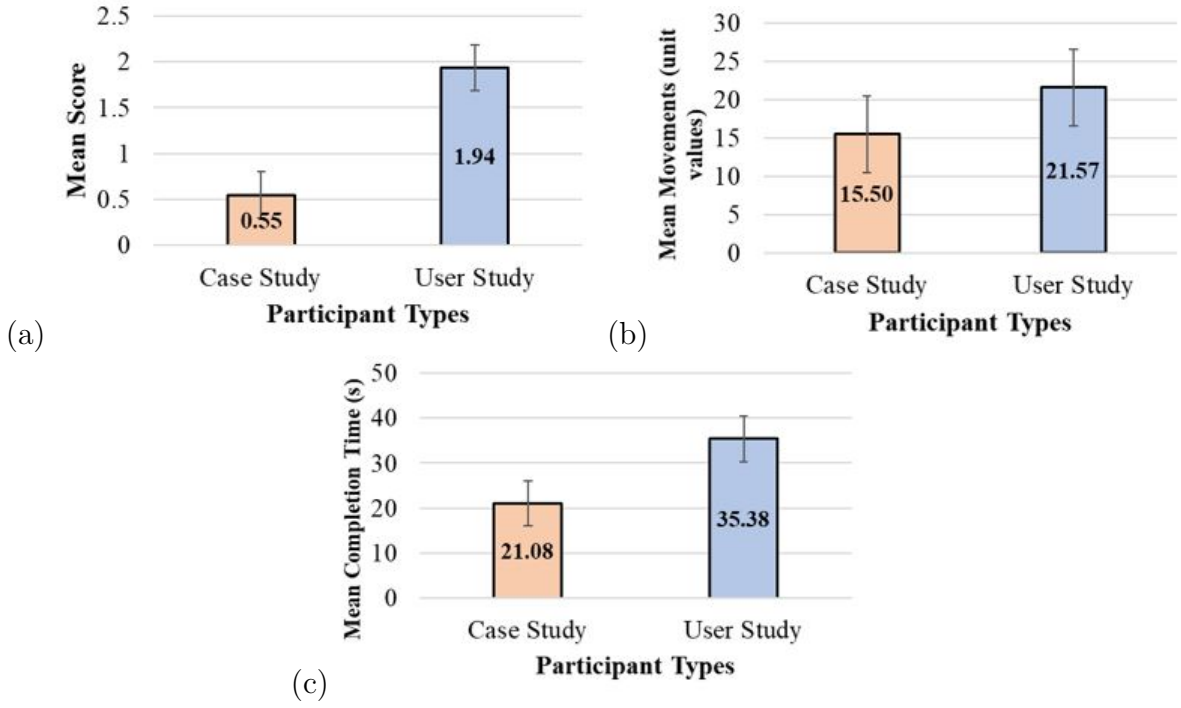
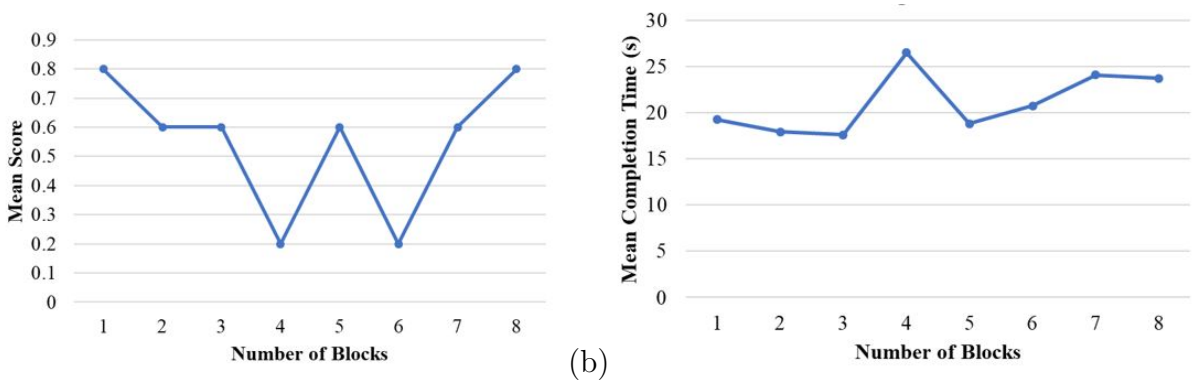


Figure 6.13: Performance measure comparisons between the case study and the user study.

Mouse experience on a five-point scale: (1 = *very poor*, 5 = *very good*), he gave *Camera Mouse* a score of 2, which equals to *poor*. He also mentioned that the sensitivity of the cursor was a challenge for him. The participant also stated that as he moved further away from the web-cam, he felt more comfortably while using the *Camera Mouse*.

6.4.4 Summary of the Case-study

We conducted this case to identify how a physically-challenged participant would perform in our hands-free gaming experiment. We followed all standard experiment procedures for this case-study. We evaluated the performance measures of this case-



(a) (b)

Figure 6.14: Learning over eight blocks with *Camera Mouse* for (a) score, and (b) completion time (s).

study with those of the user-study and found that the values of the performance measures for the case-study were lower that those of the user-study, for all cases.

Chapter 7

Conclusion

7.1 Findings from the First Experiment

Using the smart-phone without touching it is an increasingly intriguing concept – not just for recreational purposes but also to support users with a motor disability. As expected, users were comfortable while doing the text-entry using touch input, even more so during the *Qwerty* phase of the experiment. Touch input on *Opti* was new for users because of the four space buttons and the different organization of the keys.

Device tilt and head movement were significantly slower than touch. Device tilt, head movement, and touch had an entry speed of 4.66 wpm, 4.30 wpm, and 18.57 wpm, respectively. The participants could correct mistakes during text entry and, for the most part, did so: The grand mean for error rate was just 0.91%. The two non-touch methods are prone to physical fatigue. The participants pointed this out in their feedback. Although users were optimistic in their feedback when asked about the usefulness of the hands-free input system, this is simply the perspective of non-

disabled users. The device tilt method received less criticism from the users, yet it is not ideal to use such an input method in the fast-moving daily life of smart-phone users.

For physically disabled users, achieving a high entry speed is not necessarily of great importance. This user study could be extended to physically disabled users, for example, to gauge what kind of feedback they provide. The hands-free method for user input has intrigue but not the sophistication of conventional touch input. Participants pointed out that; however, it can be a useful form of interaction in the future.

7.2 Findings from the Second Experiment

In our second experiment, we compared four input methods using the 2D Fitts' law task in ISO 9241-9. The methods combined two pointing methods (touchpad, *Camera Mouse*) with two selection methods (tap, dwell). Using *Camera Mouse* with dwell-time selection is a hands-free input method and yielded a throughput of 0.65 bps. The other methods yielded throughputs of 0.85 bps (*Camera Mouse* + tap), 1.10 bps (touchpad + dwell), and 2.30 bps (touchpad + tap).

Cursor movement was erratic with *Camera Mouse*, particularly with dwell-time selection. This was in part due to the long 2000 ms dwell-time employed. Participants gave the hands-free condition a neutral, or slightly better than neutral, subjective rating.

7.3 Findings from the Third Experiment

In our third experiment, we compared *Camera Mouse* with touchpad and keyboard of a laptop computer to play a simple game: *Snake*. The keyboard was the best performing method among the three methods tested. The mean score for the keyboard was 5.89; the mean score for the touchpad was 3.34; and *Camera Mouse* had the lowest mean score at 1.94. The mean completion time for the keyboard was 40.79 seconds; the mean completion time for the touchpad method was 39.32 seconds; and *Camera Mouse* had a mean completion time of 35.38 seconds. The mean number of movements for the keyboard was 73.90; the mean number of movements for the touchpad method was 28.18; and *Camera Mouse* had a mean number of movements of 21.57. We also conducted a case study with a physically impaired participant. The performance measures showed lower values for the case study compared to the results gathered from the hands-free session.

But considering participant feedback, there are opportunities to try out *Camera Mouse* in other games as well, such as, Temple Run, Subway Surfers, point of view driving games etc.

7.4 Future Work

We have conducted each of our three experiments in a complete reproducible manner. Hence, there are opportunities to build further on these experiments and explore other dimensions of their research scope. We have identified the following as opportunities related to our experiments that can be explored from an HCI research point of view:

- Designing a hands-free text-entry experiment only for widely used phrases in

the house-hold

- Exploring random point-select tasks with *Camera Mouse*.
- Exploring other games such as Temple Run, Subway Surfers, and Point of View Driving games with *Camera Mouse*.
- Engaging more physically-challenged participants.

Voice command, gesture control, and brain-computer research are also significant forms of hands-free interaction. In future, the door remains open to explore these forms of hands-free interactions against facial tracking methods. It is also worthwhile to mention that accessible computing does not necessarily mean that a user has to be completely detached from the machine. Hence, partially hands-free methods can also be valuable for user-interaction in the accessible computing domain.

Bibliography

[1] Betke, M., Gips, J., and Fleming, P. The camera mouse: Visual tracking of body features to provide computer access for people with severe disabilities. *IEEE Transactions on neural systems and Rehabilitation Engineering 10*, 1 (2002), 1–10.

[2] Card, S. K., English, W. K., and Burr, B. J. Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a CRT. *Ergonomics 21* (1978), 601–613.

[3] Cloud, R., Betke, M., and Gips, J. Experiments with a camera-based human-computer interface system. In *Proceedings of the 7th ERCIM Workshop" User Interfaces for All," UI4ALL* (2002), ERCIM, pp. 103–110.

[4] Corcoran, P. M., Nanu, F., Petrescu, S., and Bigioi, P. Real-time eye gaze tracking for gaming design and consumer electronics systems. *IEEE Transactions on Consumer Electronics 58*, 2 (2012).

[5] Cuaresma, J., and MacKenzie, I. S. A comparison between tilt-input and facial tracking as input methods for mobile games. In *Games Media Entertainment (GEM), 2014 IEEE* (New York, NY, USA, 2014), IEEE, pp. 1–7.

[6] Fitts, P. M. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology 47* (1954), 381–391.

[7] Gips, J., Betke, M., and Fleming, P. The camera mouse: Preliminary investigation of automated visual tracking for computer access. In *In Proc. Conf. on Rehabilitation Engineering and Assistive Technology Society of North America* (2000), RESNA, pp. 98–100.

[8] Grammenos, D., Savidis, A., and Stephanidis, C. Ua-chess: A universally accessible board game. In *Universal Access in HCI: Exploring New Interaction Environments-Proc. 11th Int. Conf. on Human-Computer Interaction (HCI International 2005)* (2005), vol. 7.

[9] Gregor, P., Sloan, D., and Newell, A. F. Disability and technology: building barriers or creating opportunities? *Advances in computers 64* (2005), 283–346.

[10] ISO. Ergonomic requirements for office work with visual display terminals (VDTs) - part 9: Requirements for non-keyboard input devices (ISO 9241-9). Tech. Rep. Report Number ISO/TC 159/SC4/WG3 N147, International Organisation for Standardisation, 2000.

[11] ISO. Evaluation methods for the design of physical input devices - ISO/TC 9241-411: 2012(e). Tech. Rep. Report Number ISO/TS 9241-411:2102(E), International Organisation for Standardisation, 2012.

[12] Kaufman, A. E., Bandopadhay, A., and Shaviv, B. D. An eye tracking computer user interface. In *Virtual Reality, 1993. Proceedings., IEEE 1993 Sym-*

posium on Research Frontiers in (New York, NY, USA, 1993), IEEE, pp. 120–121.

[13] Kent, S. L. *The Ultimate History of Video Games: Volume Two: from Pong to Pokemon and beyond... the story behind the craze that touched our li ves and changed the world*. Three Rivers Press, 2010.

[14] Levenshtein, V. I. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady* (Moscow, Russia, 1966), vol. 10, pp. 707–710.

[15] MacKenzie, I. S. Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction 7* (1992), 91–139.

[16] MacKenzie, I. S. Kspc (keystrokes per character) as a characteristic of text entry techniques. In *International Conference on Mobile Human-Computer Interaction* (New York, NY, USA, 2002), Springer, pp. 195–210.

[17] MacKenzie, I. S. An eye on input: Research challenges in using the eye for computer input control. In *Proceedings of the ACM Symposium on Eye Tracking Research and Applications - ETRA 2010* (New York, 2010), ACM, pp. 11–12.

[18] MacKenzie, I. S. Evaluating eye tracking systems for computer input. In *Gaze interaction and applications of eye tracking: Advances in assistive technologies*. IGI Global, Hershey, PA, USA, 2012, pp. 205–225.

[19] MacKenzie, I. S. *Human-computer interaction: An empirical research perspective*. Morgan Kaufmann, Waltham, MA, USA, 2012.

[20] MacKenzie, I. S. Fitts' law. In *Handbook of human-computer interaction*, K. L. Norman and J. Kirakowski, Eds. Wiley, Hoboken, NJ, 2018, pp. 349–370.

[21] Magee, J., Felzer, T., and MacKenzie, I. S. Camera mouse+ clickeraid: Dwell vs. single-muscle click actuation in mouse-replacement interfaces. In *International Conference on Universal Access in Human-Computer Interaction* (2015), Springer, pp. 74–84.

[22] Magee, J. J., Scott, M. R., Waber, B. N., and Betke, M. Eyekeys: A real-time vision interface based on gaze detection from a low-grade video camera. In *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW'04. Conference on* (New York, NY, USA, 2004), IEEE, pp. 159–159.

[23] Partridge, K., Chatterjee, S., Sazawal, V., Borriello, G., and Want, R. Tilttype: accelerometer-supported text entry for very small devices. In *Proceedings of the 15th annual ACM symposium on User interface software and technology* (New Jersey, NY, USA, 2002), ACM, pp. 201–204.

[24] Roig-Maimó, M. F., Manresa-Yee, C., Varona, J., and MacKenzie, I. S. Evaluation of a mobile head-tracker interface for accessibility. In *Proceedings of the 15th International Conference on Computers Helping People With Special Needs - ICCHP 2016 (LNCS 9759)* (Berlin, 2016), Springer, pp. 449–456.

[25] Shneiderman, B. Touch screens now offer compelling uses. *IEEE software 8*, 2 (1991), 93–94.

[26] Soukoreff, R. W., and MacKenzie, I. S. Measuring errors in text entry tasks: an application of the levenshtein string distance statistic. In *CHI'01 extended abstracts on Human factors in computing systems* (New Jersey, NY, USA, 2001), ACM, pp. 319–320.

[27] Soukoreff, R. W., and MacKenzie, I. S. Towards a standard for pointing device evaluation: Perspectives on 27 years of Fitts' law research in HCI. *International Journal of Human-Computer Studies 61* (2004), 751–789.

[28] Toyama, K. Look, ma-no hands! hands-free cursor control with real-time 3d face tracking. *PUI98* (1998).

[29] Wang, J., Zhai, S., and Canny, J. Camera phone based motion sensing: interaction techniques, applications and performance study. In *Proceedings of the 19th annual ACM symposium on User interface software and technology* (New Jersey, NY, USA, 2006), ACM, pp. 101–110.

[30] Wigdor, D., and Balakrishnan, R. Tilttext: using tilt for text input to mobile phones. In *Proceedings of the 16th annual ACM symposium on User interface software and technology* (New Jersey, NY, USA, 2003), ACM, pp. 81–90.