

Institute for Visualization and Interactive Systems (VIS)
Socio-Cognitive Systems (SKS)
University of Stuttgart
Pfaffenwaldring 5a
D-70569 Stuttgart

**User-defined transfer functions
to improve pointing performance
in graphical user interfaces**

S.M. Hasanul Banna

Course of Study : INFOTECH

Examiner : Jun.-Prof. Dr. Niels Henze

Date of Submission : May 12, 2017

CR-Classification : H.5.2

Abstract

Pointing at a target is the most fundamental and frequent task in graphical user interfaces (GUIs). Pointing devices like the mouse is the most popular and cheapest input devices for current desktop computers and the touchpad or trackpad is the best match between performance and demand of pointing devices for laptop computers. Due to the widespread and frequent use of pointing devices, even a small improvement in pointing performance can have a large effect on a system's usability.

The physical movement of a mouse or dragging motion of a finger on a touchpad is translated into the movement of a pointer on the graphical display through so-called transfer functions. Transfer functions are actually the only pointing facilitation technique available to all users in modern days operating systems. Despite the importance of transfer functions, very little is known about the nature of the optimal transfer functions.

Pointer acceleration (PA) is the default behavior on the Microsoft Windows and Apple macOS operating systems. It dynamically manipulates the Control-Display (CD) gain between the input device and the pointer as a function of the device's velocity. This mechanism has been implemented in modern desktop user interfaces and increases the CD gain as the user's hand or finger velocity increases. Previous work showed that the default functions of Windows, Apple macOS and Xorg (the X.Org Foundation server) have shown better performance compared to a constant CD gain. Apple macOS has the improvised performance for small target widths but reduces performance for covering the long distances. Current knowledge on velocity based transfer functions relies on evaluations of basic functions and adapting the CD gain in discrete or continuous ways using low-order polynomials. The internal details and design rationales of the transfer functions that we all use are mostly unknown.

The aim of this thesis is to gain a deeper understanding of the optimal transfer functions and to assess them based on natural interaction with the system using a user-driven approach. The implementation part of the thesis is divided into two desktop applications, one is for recording all raw mouse and pointer movement as well as recording contextual information to understand a transfer function. The other implementation part of the thesis is to enable the user to define their own transfer functions. Users can customize existing default transfer functions and use them to control the pointer. These two applications are used to conduct a study that collects user device information and as well as user-defined transfer functions. Finally, we identify interesting transfer function for touchpads and mice.

Acknowledgement

Foremost, I would like to thank Jun.-Prof. Dr. Niels Henze for not only giving me the opportunity to do my master's thesis in the Socio-Cognitive Systems group of the Institute for Visualization and Interactive Systems, University of Stuttgart but also his enthusiasm, wisdom, cordial supervision and valuable advice that helped me a lot throughout my thesis. I am earnestly grateful to my professor for the fruitful discussions, providing significant guidelines for solving difficult problems in an easier way, important corrections throughout the writing of my thesis and educating me how to be able to write.

I also want to thank the people who work at this department and who provided instant support for any difficulty I faced.

I am very grateful to my father and mother for the continuous support starting from the childhood to today. If not for their encouragement, care, love, passion, and their assistance, I would not be here writing my master's thesis. I am blessed to have my beautiful wife and princess she is carrying, despite her pregnancy she supported me to share my stress during the thesis period. I sincerely appreciate her support and sacrifice.

Also, I would like to thank all survey participants and users of the application for their willingness to participate in the survey and spend their time for the feedback.

Finally, I would like to finish by expressing my acknowledgment and love to all of my friends for their support in all the ways during my master's study.

Contents

1	Introduction	9
2	Related Work and Background	13
2.1	Introduction	13
2.2	Pointing Devices	14
2.2.1	Motion-tracking pointing devices	15
2.2.1.1	Mouse	15
2.2.2	Position-tracking pointing devices	16
2.2.2.1	Touchpad	16
2.2.2.2	Touchscreen	16
2.2.3	Pressure-tracking pointing devices	17
2.2.3.1	Isometric Joystick	17
2.3	Fitt's Law	17
2.4	Constant Control Display Gain	18
2.5	Pointer Acceleration	19
2.6	Transfer Functions	20
2.6.1	Windows Transfer Function	20
2.6.2	X.Org Foundation Server Transfer Functions	24
2.6.3	Apple OS X Transfer Functions	26
2.7	libpointig toolkit and EchoMouse	28
2.7.1	EchoMouse	28
2.7.2	libpointig	29
2.8	Summary	30
3	Concept of User-defined Transfer Function	31
3.1	Introduction	31
3.2	Mouse Pointer Recorder	32
3.3	User-defined Application Tool	33
3.3.1	Defaults Transfer Function	34
3.3.2	Interactive Graph	34
3.4	Summary	34
4	Design and Implementation	37
4.1	Introduction	37
4.2	Mouse Logger	37

4.2.1	Architecture and Requirements	37
4.2.2	Design	38
4.2.3	Implementation	39
4.2.4	Log File	40
4.3	User-defined Transfer Function Tool	41
4.3.1	Architecture and Requirements	41
4.3.2	Design	43
4.3.3	Implementation	44
4.3.3.1	Notification and System Tray Icon	44
4.3.3.2	Load libpointing Data Points	44
4.3.3.3	Inside Calculation	45
4.3.4	Log File	48
4.4	Summary	48
5	Study for Interesting Transfer Function	49
5.1	Introduction	49
5.2	Design Deployment	49
5.2.1	Extended Graphical User Interface	50
5.2.2	Log File	51
5.2.3	Google Sheets	53
5.3	Participants	54
5.4	Apparatus and Materials	55
5.5	Procedure	56
5.6	Result	57
5.7	Discussion	61
5.8	Summary	67
6	Conclusion and Future Work	69
	Bibliography	71
	List of Figures	77
	List of Tables	79
	Abbreviations	81
A	Appendix	83
A.1	Log File Screen Shot	83
A.2	Class Diagram	86
A.3	Google Sheets	89
A.4	libpointing data points	93
A.5	Trials and Feedbacks Activated Duration.	95
A.6	Interesting Transfer Function for Individual Participants.	96
A.7	Box Plot Diagram.	102
A.8	Interesting Transfer Functions.	105

1 | Introduction

Since the invention of graphical displays in the fifties-sixties, pointing at a target has become the fundamental and frequent activities. Indirect control of a pointer on the display, with a separate device, has been the most common mechanism of pointing in *graphical user interfaces* (GUIs). After the appearance of *graphical user interfaces*, the first pointing devices like the light pen, joystick, and mouse were introduced and lead to more intuitive access of electrical resources.

Pointing devices can be classified according to the way an on-screen pointer is controlled. Examples include according to a change of device's movement,controlling, positioning or resistance. Among all pointing devices, the mouse is the most popular pointing device for desktop computers. As explained by Moggridge and Atkinson [25], not only Douglas Engelbart and his colleagues (W. K. English, M. L. Berman) invented it but also confirmed that it works best compared to a range of alternative devices. Fifty years later, the mouse still provides a good match between human performance and the demands of graphical desktop interfaces [13]. Considering the pointing performance and usability, touchpad also offers a similar match for a laptop computer.

The physical movement of a mouse is translated into the movement of a pointer on the graphical display through so-called *transfer functions*. The goal of the function is to improve pointing performance by providing more stable and intuitive control over devices. A *transfer function* that matches the properties of an input device is known as an appropriate mapping. For force sensing input devices, such as joysticks and trackpoints, the transfer function should be a force-to-velocity function. Other appropriate mappings include position-to-position for the touchscreens of smartphones and tablets, as well as velocity-to-velocity functions for mice. The important fact is that the relation between pointer and device has to be hardware - independent and is described using standard length and time units (e.g. meters).

These transfer functions can be divided into three types, *constant gain (linear)*, *discrete gain increase (discrete non-linear)* and *continuous gain (continuous non-linear)* [9]. For *linear transfer functions*, the term *Control Display* (CD) gain is commonly used to refer to the scale factor between the pointing control device and the visual pointer. For *linear transfer functions*, the CD gain is constant but *nonlinear transfer functions* dynamically manipulates the CD gain as a function of the device's velocity and is called *Pointer Acceleration* (PA).

When the velocity of the control device is high the CD gain is high and when the velocity of the control device is low the CD gain is low. PA is the default behavior on the Microsoft Windows XP/Vista and macOS operating systems. Discrete gain change showed no advantage in overall movement time over constant gain, which was the finding of Jellinek and Card [15]. Moreover, continuous gain significantly degraded (increased) movement time. This degradation in the nonlinear condition was also seen as an increase in the average number of corrective submovements and increased in effective target width compared with the constant gain condition. Casiez et al. [7] found that pointing acceleration has a small performance advantage over constant CD gain when selecting small targets or covering long distances.

Casiez and Roussel [6] conducted an experiment to describe and compare the *transfer functions* of the different operating system. Their results showed that PA functions had improved performance compared to a constant CD gain. They also found that OS X had improved performance for small target widths but reduced performance for larger ones compared to Windows and Xorg. The *transfer functions* of Microsoft Windows, Apple OS X and Xorg (the X.Org Foundation server) are actually the only pointing facilitation technique available to all users of these systems. But despite the importance of *transfer functions*, very little is known about the nature of an optimal transfer function and the internal details and design rationales are mostly unknown.

This thesis will start with a detailed review of the related work, explaining how pointing facilitation research deals with optimal *transfer functions* and what is known about them. Then it assesses the *transfer functions* based on natural interaction with the system, using a user-driven approach. In the implementation part of the thesis, we have developed two desktop application tools, one of them records pointing device and cursor information and another to enable users to define their own *transfer functions*. By using this developed desktop application tools, a study has been conducted during participants daily work to identify interesting use-defined *transfer functions* for mouse and touchpad pointing devices.

Outline

The thesis is structured as follows:

Chapter 2 - Related Work and Background: In this chapter, we discuss basic and background information regarding pointing devices and transfer functions. We also present related work in the field of transfer functions according to different pointing devices and operating systems.

Chapter 3 - Concept of User-defined Transfer Functions: Based on the previous chapter in this chapter, we present the concept for user-defined transfer functions, the idea of desktop application tools and the concept of the user study.

Chapter 4 - Design and Implementation: In this chapter, we describe details implementation of universal desktop mouse logger and user-defined desktop application tool.

Chapter 5 - Study for Interesting Transfer Function: In this chapter, we describe the deployment of the developed application for user study and details description of the user study procedure. After that, present the user study result, analyze the collected data and identify interesting transfer function.

Chapter 6 - Conclusion and Future Work: In this chapter, we summarize conclusions that we can draw from our work and discuss future work.

2 | Related Work and Background

2.1 Introduction

Pointing *transfer function* is the general mechanism for pointing facilitation technique. This technique is used all of the modern operating systems. Millions of people around the world used this transfer function on a daily basis. To understand this pointing mechanism and technique, it is important to look at the prior researches on other pointing facilitation technique and pointing devices.

Transfer functions are generally considered as control variables, that means they may influence dependent variables or hold constant from one condition to another. But there is no clear description of these control variables. However, the prior literature on pointing facilitation shows that the details are often incomplete and unclear.

Casiez and Roussel [6] conducted a controlled experiments to characterize and compare pointing transfer functions. In their experiment, they found that default transfer function used in Windows, macOS and Xorg outperformed a constant CD gain, but they had also accepted that using constant CD gain function as a baseline to compare with other techniques should have been prohibited unless clearly justified.

Unfortunately, many authors do not provide all the details while using a constant CD gain or ratio as baseline or base technique. Their research on pointing facilitation and the way of disabling the operating system's transfer functions during their work were not precisely characterized [6]. In the MacKenzie and Isokoski [21] research, there is no way of knowing which transfer function was used, only written "*an optical USB Microsoft IntelliMouse with four buttons and a scroll wheel ... an experimental software written in Java*".

Enforcing a hardware-independent transfer function even a constant gain is very difficult with the current system. Wobbrock et al. [30] in Angle Mouse study has acknowledged that "*although some on-line documentation discusses pointer ballistics in Windows, it does not contain sufficient information to establish the slider-to-gain mapping*".

Most of the modern pointing devices conform to the *Human Interface Devices* (HID)

class of the USB standard. This class comes with a variety of equipment including keyboards, mice, touchpads, joysticks, remote controls, barcode readers and voltmeters. Among other devices, a pointing device description specifies for each axis whether transmitted values are absolute or relative, linear or nonlinear, their byte size, their logical range, the corresponding physical range and the unit system and exponent used, also specifies the polling rate and polling intervals [3].

This chapter is organized as follows. First, we describe popular pointing devices according to the common classification and basic terms. After that, we describe details about most popular transfer functions in the modern operating systems.

2.2 Pointing Devices

A pointing device is an input human interface that allows a user to input continuous or multi-dimensional data to a graphical user interface. It allows the user to control and provide data to the graphical system using physical gestures like point, click, drag and drop. The pointing devices can be divided into several characteristics like direct vs. indirect input, absolute vs. relative movement, isotonic vs. elastic vs. isometric, position control vs. rate control and degrees of freedom.

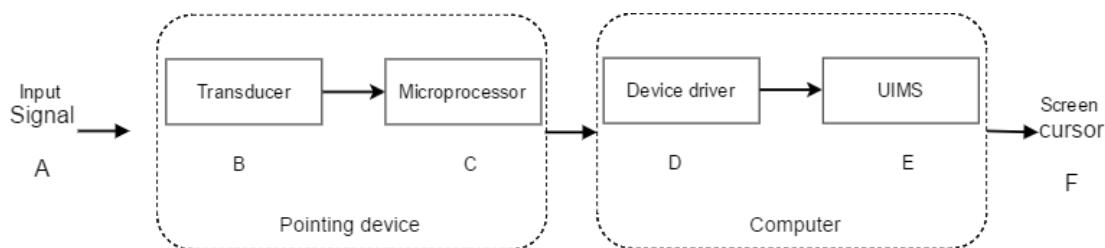


Figure 2.1: Generic block diagram of a pointing device (from [9]).

The main concept of pointing devices is the transformation of information sensed by a physical device into the movement of the pointer on the visual screen. The figure 2.1 shows the generic block diagram of a pointing device, the user manipulates a physical property of the device (A), such as position which is sensed by the transducer (B) and input as changes in an electrical quantity, such as voltage, into a device microprocessor (C). In most modern pointing devices, the microprocessor translates the signal from analog to digital by using an A-D converter. The transducer and the microprocessor are usually packaged as part of the device, The microprocessor may also compute a sophisticated transfer function which takes the digitized signal and transforms it into displacement or velocity data. The pointing device is usually connected to the main computer through a standardized connection called a *Port* [9].

The microprocessor generates interrupts to signal that new data is available from the pointing device. This data is read by specialized software in the main computer

known as the device driver (D), which generates pointing device events to that part of the operating system responsible for graphics and windows management. This is called the user-interface management system (UIMS) (E). The UIMS is responsible for the graphics which finally creates cursor position and motion on screen (F) [9].

Pointing devices can be classified into many classes but the most common classification are motion, position and pressure tracking pointing devices. Few most popular pointing devices from each classification are listing below.

2.2.1 Motion-tracking pointing devices

2.2.1.1 Mouse

Pointing devices which track the physical motion of the input device and translate it into the virtual motion on the screen. The most common motion tracking pointing device is the mouse. A computer mouse is an indirect, relative, isotonic, position-control, translational input device with two degrees of freedom and three states.

First Mouse, The inventors are overall recognized to be Douglas Engelbart and Bill English. They presented their work in the year 1965, at the Stanford Research Institute (Menlo park, California) [10]. The purpose of the research was one of the first studies about human and computer interaction, in which pointing input devices were confronted in order to find a way to simplify and improve computer operation. Their solution to point and select on the computer screen is considered to be the first mouse. The first mouse was a wooden case of about 5x7.5x10 cm. The first mouse had two potentiometers which were perpendicularly placed and were responsible for the movements in horizontal and vertical directions, with a button to select text.

Ball Mouse, while the research on the first mouse was taking place, a European evolution of the mouse appeared during the development of the TR440 Computer [16]: the ball mouse, that simplified drawing simple vector graphics on the screen, such as simple geometrical figures. In older mouse version the movements were restricted to horizontal or vertical, the adoption of the ball instead brought more liberty in the movements and permitted the user to move the mouse in any direction. This ball-mechanical mouse and particularly the *Xerox-Model* designed in the seventies will be a fundamental part of commercial mice in the nineties.

Lisa Mouse, this Apple's mouse is the first wide-used computer pointing-input device, in this mouse, there is no substantial difference from the Xerox's model, except the cost 20 times less. This reduction of cost is an important step in the diffusion of mice, thus it is bought by much more people than before during the eighties.

Optical Mouse, the first optical mouse had been developed in 1981 at Palo Alto from Richard F. Lyon [19]. The working principle of an optical mouse is simple, a chip takes photos at every small step of time, and then compares the images to recognize where the mouse is moving. Due to the use of this technology, an optical mouse does not work on perfect or reflecting surfaces. An optical mouse needs some pattern on the table to reconstruct the movement. A popular belief is that all optical mice are laser mice. This is not the case: most of the optical mice are based on an infrared LED and this perfectly explains the red light on the bottom of the mouse. The most important advantages are that this device needs less maintenance and has the higher lifetime, due to the fact that the mechanism is isolated, so that dust and dirt cannot enter into the mouse. It is also lighter and cheaper than a ball mouse.

2.2.2 Position-tracking pointing devices

2.2.2.1 Touchpad

A touchpad is an indirect, absolute, isometric, position-track input device with two degrees of freedom and two states and a touch screen is a direct, absolute, isometric, position-control input device with two degrees of freedom and two states. The touchpad provides a competitive match with mouse and satisfies the need of human performance and the demands of graphical user interfaces for the laptop. A touchpad or trackpad is a flat surface that can detect finger contact. It is a stationary pointing device. At least one physical button normally comes with the touchpad, but the user can also generate a mouse click by tapping on the pad. Advanced features include pressure sensitivity and special gestures such as scrolling by moving one's finger along an edge. It uses a two-layer grid of electrodes to measure finger movement: one layer has vertical electrode strips that handle vertical movement, and the other layer has horizontal electrode strips to handle horizontal movements [14].

2.2.2.2 Touchscreen

A touchscreen is a display device that allows the user to interact with a computer by using their finger or some other helping instruments. The touchscreens are embedded into the screen of the graphical display devices such as computer's monitors and smart gadgets. Users interact with the device by physically pressing items shown on the screen, either with their fingers or some helping tool.

Resistive Touchscreens are the most common touchscreen technology. They are used in high-traffic applications and are immune to water or other debris on the screen. Resistive touchscreens are usually the lowest cost touchscreen implementation. Because they react to pressure, they can be activated by a finger, gloved hand, stylus or other objects like a credit card.

Surface Capacitive Touchscreens provide a much clearer display than the plastic cover typically used in a resistive touchscreen. In a surface capacitive display, sensors in the four corners of the display detect capacitance changes due to touch. These touchscreens can only be activated by a finger or other conductive objects.

Projected Capacitive Touchscreens are the latest entry to the market. This technology also offers superior optical clarity, but it has significant advantages over surface capacitive screens. Projected capacitive sensors require no positional calibration and provide much higher positional accuracy. Projected capacitive touchscreens are also very exciting because they can detect multiple touches simultaneously [18].

2.2.3 Pressure-tracking pointing devices

2.2.3.1 Isometric Joystick

An isometric joystick is an indirect, relative, elastic, rate-control, translational input device with two degrees of freedom and two states. In contrast to a 3D Joystick, the stick itself doesn't move or just moves very little and is mounted in the device chassis. To move the pointer, the user has to apply force to the stick. Typical representatives can be found on notebook's keyboards between the "G" and "H" keys. By performing pressure on the TrackPoint, the cursor moves on the display [29].

There are many more devices that have been developed in different forms to catch particular needs. For example, a joystick is a perfect choice for a car racing game, but it is not good for pointing. The touchpad or trackpad are not as intuitive as touchscreens. The cost is the matter for most of the devices to perform more quickly and instinctively. By far mouse is the most common and popular pointing device for graphical user interface.

2.3 Fitt's Law

Pointing device movements are very important for operating a graphical user interface. The performance of the pointing devices is often measured by the target hit completion time and accuracy. In a common scenario, if the target is further away or smaller then it takes more time to hit. Fitts' [11] research allows predicting the time a human needs to point at a target of given size in a given distance. Fitts' law states that it takes more time to hit a target if the target is further away and it also takes more time if the target is smaller. Fitts' law also states that the target acquisition time increases drastically if the target gets tiny. It is clear that an infinite small target is impossible to hit, means it takes infinite time.

Fitts' Law is a very successful experimental paradigm that has been widely applied to the comparison, optimization and measuring the effectiveness and accuracy of

a pointing device. Fitts' Law was first applied to the study of input devices by Card et al. [5]; it is now a standard for device comparisons. Originally it is used to model direct pointing where the hand taps physical objects. Fitts' law is also successfully used for indirect pointing where the control device and display pointer are decoupled [21, 20]. The decoupling of control and display creates two different spaces: the display space, where view a representation of the pointing action, and the motor space, where to manipulate the control device. Given the intended target's width W and distance D , the total movement time T is predicted with the following equation using MacKenzie's[20] Shannon formulation:

$$T = a + b \log_2 \left(\frac{D}{W} + 1 \right) \quad (2.1)$$

The constants a and b are empirically determined for the pointing technique and/or device being used. The formulation of Fitts's index of difficulty (ID) most frequently used in the Human-Computer Interaction community is:

$$ID = \log_2 \left(\frac{D}{W} + 1 \right) \quad (2.2)$$

Intuitively it shows that tasks become more difficult as a target moves farther away, or as a target becomes smaller.

2.4 Constant Control Display Gain

The Control-Display (CD) Gain describes the unit free proportion that maps the movements between in the control pointing device to the movements of the display pointer [12]. The reciprocal is called the CD ratio [27]; For example, a hardware mouse moves in another speed or distance than the cursor moves on the screen and the measurement units have to be same in order to be meaningful (e.g. meters instead of pixels). The CD gain refers to the scale factor of these two movements:

$$CD_{gain} = \frac{V_{pointer}}{V_{device}} \quad (2.3)$$

If CD gain is 1, the display pointer moves at exactly the same distance and speed as the control device; when CD gain is greater than 1, the display pointer moves proportionality further and faster than the control device; and when CD gain is less than 1, the display pointer moves slower, covering less distance than the control device. The CD gain can be computed by taking the ratio of the pointer velocity to device velocity (equation 2.3).

Quantization can become a problem if the maximum resolution of the control device together with a high CD gain prevents every pixel from being addressable on the

display. The maximum CD gain that can be used without quantization problems is calculated by dividing the resolution of the pointing device by the resolution of the display using the same unit of measurement (e.g., DPI).

When CD gain is very low and/or the physical device movement area is constrained, the device may need to be clutched to move the display pointer over a long distance. *Clutching* is when a device is repositioned in motor space without affecting the display pointer. The device movement area constraint may be a well-defined characteristic of the device, such as the limited input area on laptop trackpads, or less defined, such as the comfortable range of arm movement or unobstructed surface space. The maximum area of unconstrained physical movement is called operating range of the device.

However, a compromise has to be found with high gains it is easier to approach a distant target, with low gains it takes longer. High gains hinder the selection of targets, whereas low gains facilitate this process [24]. The operating systems like Microsoft Windows, OS X and Xorg have implemented mechanisms in order to adapt the CD gain to the user's needs, e.g. the CD gain increases when the user's movement velocity increases [6].

2.5 Pointer Acceleration

Pointer Acceleration (PA) dynamically increases CD gain as the velocity of the control device increases. This behavior is motivated by the *optimized initial impulse* motor control model which is a hybrid of the *iterative corrections model* and the *impulse variability model* [24].

The *iterative corrections model* [8, 17], attributes the law entirely to closed-loop feedback control. This model states that the whole movement consists of a series of discrete sub-movements, each of which takes the user closer to the target and is triggered by feedback indicating the target which is not yet attained.

And the *impulse variability model* [28], attributes the law almost entirely to an initial impulse delivered by the muscles, flinging the limb towards the target. The last part of the movement time consists of the limb merely coasting towards the target.

It works as follows: an initial high-velocity ballistic movement is made in the direction of the target. If the ballistic movement ends on the target, the task is complete, but if the movement undershoots or overshoots the target, a second lower velocity corrective movement is used in the direction of the target. Successively slower corrective movements are reapplied until the target is acquired (see figure 2.2).

Pointer Acceleration function f produces a CD gain G from the device motor space velocity v [7].

$$G = f(v) \tag{2.4}$$

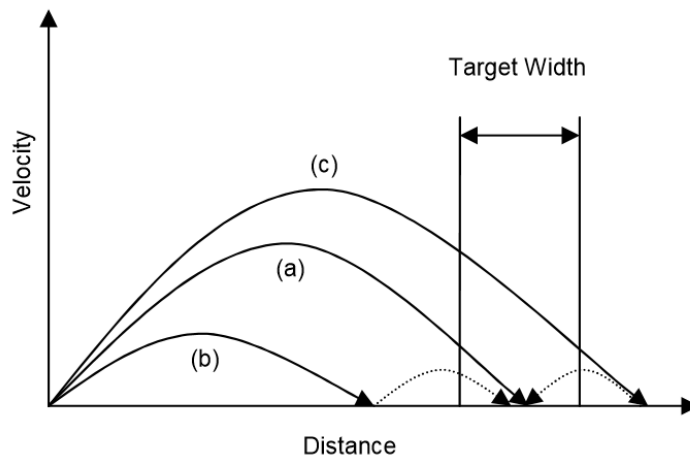


Figure 2.2: Possible sequence(s) of submovements toward a target as described by the optimized initial impulse model [26]. (a) is the case where a single movement reaches the target. (b) and (c) are the more likely cases where the initial movement under or over shoots the target, requiring subsequent corrective movements (from [23]).

Pointer Acceleration is one of many techniques that influences the motor space through which the device travels during target acquisition: High gain reduces the motor distance during ballistic movement, and low gain increases the motor size of the target during corrective action.

2.6 Transfer Functions

When the physical movement of the pointing devices is translated into the movement of the pointer or cursor on the graphical user interface through a function. The transfer function is a mathematical transformation that scales the data from an input device and generates movements on screen. The general goal is to provide more stable and more intuitive control of input device. A transfer function that matches the properties of an input device is known as an appropriate mapping.

Here we will discuss most popular operating systems transfer functions such as Windows, Xorg and macOS respectively:

2.6.1 Windows Transfer Function

Mouse acceleration in an operating system is an overlooked feature. But Microsoft has introduced an effective design called the *enhanced pointer precision*. Without the mouse data being altered in any way, the speed of the pointer is too high for cover long distance but at the same time too low for effective pixel-precise navigation. No matter what the mouse DPI(Dot per Inch) is, with the slider scales pointer speed

linearly increase or decrease. Microsoft has developed a transfer curve, in which slow mouse movement would result in even slower pointer movement, *subpixelation*, which allows every pixel to be pointed. Simultaneously, as the mouse speed increases, the pointer speed would also increase according to the curve.

Windows (XP) had defined a transfer function that relates the actual velocity of the mouse to the actual velocity of the pointer on the screen. Then an algorithm was applied to that transfer function to calculate the transferred pointer data. Arbitrary units of the mouse were transformed to physical units by taking the X or Y value of the mouse called mickey and scaling it using the update rate of the mouse bus and by the resolution of the pointing device [1]:

$$V_{mouse} = mickey * \frac{BusUpdateRate}{PointerResolution} \quad (2.5)$$

The typical bus update rate for a USB mouse is 125 Hz, and the typical pointer resolution is 400 mickey/inch. The typical range of a mickey size per packet that coming from a mouse is between 0 and +50, though +127 is allowed in the packet structure. To transfer the screen from arbitrary units to physical units the following relationship was used [1] :

$$V_{pointer} = mickey * \frac{ScreenUpdateRate}{ScreenResolution} \quad (2.6)$$

For example, a typical 17-inch monitor running at 1024 X 769 will have a resolution of about 80 DPI and a refresh rate of about 75 Hz. The physical velocity of the pointer on the screen is three times faster than the physical velocity of the mouse.

When the physical units were established, the parent transfer function is constructed based on a usability study. The parent transfer function is illustrated in the graph on figure 2.3 (left).

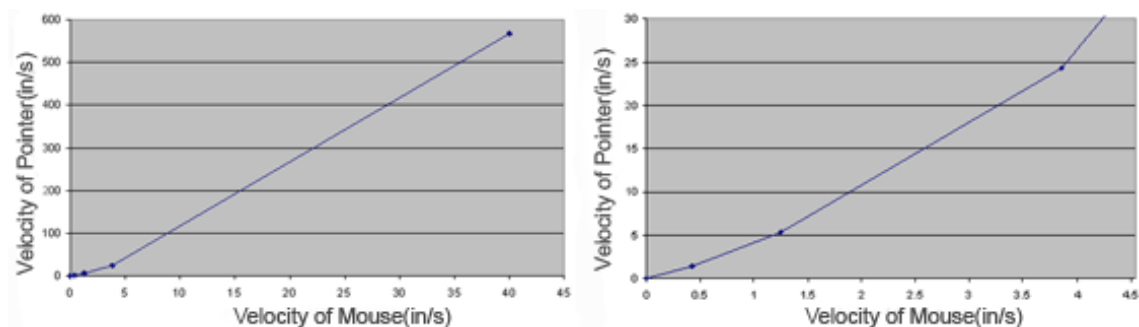


Figure 2.3: Parent transfer function graph of Windows XP with five points, and zoom view of four points are on the right side(from [1]).

The transfer function consists of five points. Four of the five points reside at the lower end of the mouse velocity spectrum (figure 2.3[right]). The velocities that go

2. Related Work and Background

beyond the 4-inch limit are linearly extrapolated. These five coordinate pairs or inflection points which form the acceleration curve are stored in the registry. These registry keys are named *SmoothMouseYCurve* and *SmoothMouseXCurve*. The first inflection point is always $(0, 0)$. The curve is linearly extrapolated beyond the last inflection point, which is to happen rarely as the physical velocity of the mouse has to be over 40 inches per second [1].

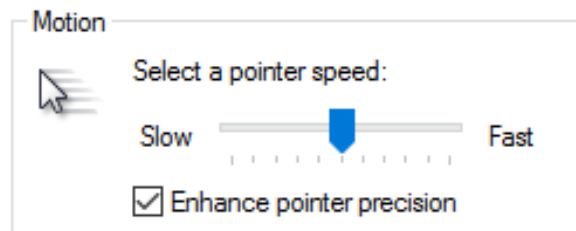


Figure 2.4: Windows 10 configuration interface with default settings. Same setting used by other old version of windows.

The slope of the first line segment yields a physical-to-screen gain of less than one gain, providing the precision movement and ability to target every pixel on the screen, or *subpixelation*. *Subpixelation* is when the user has to move the mouse physically farther than the pointer moves on the screen, by giving a high degree of precision and stability at low velocities. To achieve *subpixelation*, the divided remainder mouse counts have to be preserved and added to the next counts. The most important feature to note is that regardless of whether a fast or slow curve is selected, the first set of points tends towards *subpixelation*, which means that the user is always able to target every pixel on the screen regardless of the mouse speed setting.

There are varieties of curves are extrapolated from the parent curve to yield a transfer function with varying speed and acceleration properties. The user selects one of these curves using the pointer speed slider in the mouse properties dialog box (*Pointer Options tab*).

The transfer function is stored as a lookup table (figure 2.5). The points between the stored values are interpolated. The number used to look up the mouse X and Y raw input translated values are the vector magnitude of incoming X and Y magnitudes. The Windows XP ballistics required the use of division with a remainder, fixed-point (16.16) integer (*SmoothMouseXCurve* and *SmoothMouseYCurve*) math was used. This is important for the *subpixelation* and the increased smoothness of the pointer movement. Therefore, the maximum resultant number from two products is $2^{16}(65536)$. While an overflow is possible. If an overflow ever becomes a problem in the future, the fixed point constants in the ballistics code are easily changed to support a 20.12 fixed-point format [1].

The following list summarizes the ballistic algorithm used in Windows XP from archived paper '*Windows Hardware Developer Center*' [1], in sequence:

Name	Type	Data
(Default)	REG_SZ	(value not set)
ActiveWindowTracking	REG_DWORD	0x00000000 (0)
Beep	REG_SZ	No
DoubleClickHeight	REG_SZ	4
DoubleClickSpeed	REG_SZ	480
DoubleClickWidth	REG_SZ	4
ExtendedSounds	REG_SZ	No
MouseHoverHeight	REG_SZ	4
MouseHoverTime	REG_SZ	400
MouseHoverWidth	REG_SZ	4
MouseSensitivity	REG_SZ	10
MouseSpeed	REG_SZ	1
MouseThreshold1	REG_SZ	6
MouseThreshold2	REG_SZ	10
MouseTrails	REG_SZ	0
SmoothMouseXCurve	REG_BINARY	00 00 00 00 00 00 00 00 15 6e 00 00 00 00 00 00 40 01 00 00 00 00 ...
SmoothMouseYCurve	REG_BINARY	00 00 00 00 00 00 00 00 fd 11 01 00 00 00 00 00 00 24 04 00 00 00 00 ...
SnapToDefaultButton	REG_SZ	0
SwapMouseButtons	REG_SZ	0

Figure 2.5: Registry Editor- Mouse Lookup Table, Microsoft Windows 10; Version 1607, 2016.

1. When the system is started or the mouse speed setting is changed, the translation table is recalculated and stored. The parent values are stored in the registry (Figure 2.5) and in physical units that are now converted to virtual units by scaling them based on system parameters: screen refresh rate, screen resolution, default values of the mouse refresh rate (USB 125 Hz), and default mouse resolution (400 dpi). Then the curves are speed-scaled based on the pointer slider speed setting in the *Mouse Properties* dialog box (*Pointer Options* tab) (see Figure 2.4).
2. Incoming mouse X and Y values are first converted to fixed-point 16.16 format.
3. The magnitude of the X and Y values are calculated and used to look up the acceleration value in the lookup table.
4. The lookup table consists of six points (the first is $[0,0]$). Each point represents an inflection point, and the lookup value typically resides between the inflections points, so the acceleration multiplier value is interpolated.
5. The remainder from the previous calculation is added to both X and Y, and then the acceleration multiplier is applied to transform the values. The remainder is stored to be added to the next incoming values, which is how *subpixonilation* is enabled.
6. The values are sent on to move the pointer.
7. If the feature is turned off (by clearing the Enhance pointer precision check box underneath the mouse speed slider in the Mouse Properties dialog box [Pointer Options tab]), the system works as it did before without acceleration.

2. Related Work and Background

All these functions are bypassed, and the system takes the raw mouse values and multiplies them by a scalar set based on the speed slider setting.

When the pointer precision is turned off, if the mouse sends data that it has been moved for example one unit to the right, the pointer speed slider at 6/11 (= no scaling) the pointer will also move one unit to the right. This could be called as *one-to-one* or 100% speed. Lower speeds (subpixelation) is achieved by delaying the pointer movement until the mouse has sent enough units to the given direction. At 50% speed (scaling multiplier 0.5, pointer speed slider at 4/11), the mouse has to send 2 counts to the right before the pointer moves one count. Here's a list of the effective scaling multipliers for each pointer speed slider setting:

Pointer Speed	Mouse Sensitivity	Position Multiplier Enhance pointer precision Off	Position Multiplier Enhance pointer precision On
1	1	0.03125	0.1
2	2	0.0625	0.2
3	4	0.25	0.4
4	6	0.50	0.6
5	8	0.75	0.8
6	10	1.00	1.0
7	12	1.50	1.2
8	14	2.00	1.4
9	16	2.50	1.6
10	18	3.00	1.8
11	20	3.50	2.0

Table 2.1: Windows mouse sensitivity when position multiplier enhance pointer precision is off & on

In this above table default value, 6 means one pixel for one count, a 800 DPI mouse at a 0.5 scaling multiplier behaves similarly to a 400 DPI mouse at 1.0 multiplier.

2.6.2 X.Org Foundation Server Transfer Functions

The pointer acceleration mechanisms currently used by Xorg were introduced in 2008. The source code for these mechanisms is publicly available as part of the Xorg source tree¹ and documentation for the design rationales and operating principles are also available², although a bit mystical.

The changes introduced by Xorg in 2008 notably aimed at facilitating the exploration of transfer functions. The current architecture of the code supports 9 different profiles implemented within the new *predictable* scheme and the older *lightweight* scheme *retained mostly for embedded scenarios*. The profiles can be considered as

¹<https://cgit.freedesktop.org/xorg/xserver/tree/>

²<https://www.x.org/wiki/Development/Documentation/PointerAcceleration/>

different transfer functions, although they share some common mechanisms and code. Numerous configuration settings are associated with them but genericity and flexibility have a price: not only is the Xorg code for pointer acceleration is much larger than the one used on other systems but also it is far less readable [6].

The *predictable* scheme computes the euclidean distance corresponding to each displacement reported by the device and divides it by the time elapsed since the previous one. This instantaneous velocity is stored in a short history list ($n = 16$ by default) and it is used to maintain a better estimation of the real pointing device velocity. Two adjustable settings also play an important part: acceleration, given as a fraction, and threshold. The first one defines a high value for the (naive) CD gain to be applied for displacements, considering a default low value of 1. The second one defines the minimum velocity that needs to be achieved to switch from the low gain to the high one. The active profile specifies how the estimated velocity will be used to determine the actual CD gain within these constraints. All computations are made with floating-point arithmetic. Remainders are preserved and never cleared [6].



Figure 2.6: Ubuntu 12.04 mouse pointer configuration interface, same setting used by others version.

Figure 2.6 shows the configuration interface available in the *Pointer speed* section of the *Mouse preferences* application of Ubuntu 12.04. A help page says about the first slider:

“Use the slider to specify the speed at which your mouse pointer will move on your screen when you move your mouse”. About the second: “Use the slider to specify how sensitive your mouse pointer will be to the movements of your mouse”.

The default profile used by Ubuntu (*classic*) and the relevant settings that can be adjusted through this particular interface.

When the threshold is non-null, the *classic* profile implements a smooth transition between the low and high gain values. The sliders shown in figure 2.6 only allow such configurations. As the label indicates, the upper slider controls the acceleration setting. When dragged, it feels like a continuous control but actually supports only a predefined set of values:

3/10(<i>slow</i>)	4/10	5/10	6/10	7/10	8/10	9/10	10/10	1/1	3/2	2/1(<i>default</i>)
	5/2	3/1	1/1	3/2	7/2	4/1	9/2	5/1	11/2	6/1(<i>fast</i>)

The bottom slider controls the threshold and actually feels like a discrete control. The available values are:

2. Related Work and Background

1 (<i>low</i>)	2	3	4 (<i>default</i>)	5	6	7	8	9	10 (<i>high</i>)
------------------	---	---	----------------------	---	---	---	---	---	--------------------

In total, the interface shown in figure 2.6 thus gives access to $19 * 10 = 190$ configurations of the *classic* profile.

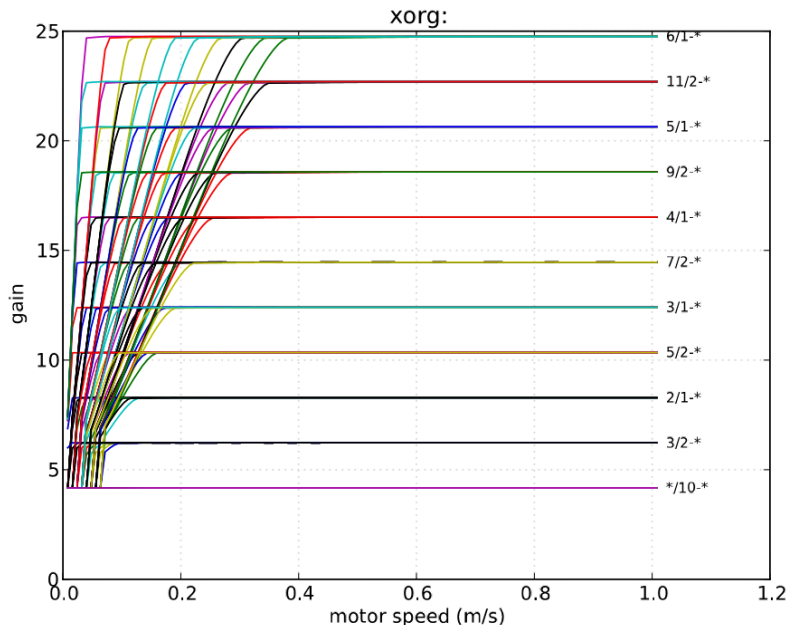


Figure 2.7: Xorg functions available in Ubuntu 10.10 through the interface shown in Figure 2.6 (from [6]).

Figure 2.7 shows a plot of these 190 functions. As one would expect, the 90 functions with an acceleration setting lesser or equal than 1, those labeled **/10-**, correspond to a naive constant gain of 1 (considering the 400 CPI and 96 PPI used for plotting the curves). Note that this is the only naive constant gain achievable through the interface is shown in figure 2.6 and that this interface does not allow to achieve a unitless constant gain [6].

2.6.3 Apple OS X Transfer Functions

The source code for the internal parts of OS X that deal with pointing transfer functions is publicly available as part of the *IOHIDFamily project*³, the main concerned files being *IOHIDSystem/IOHIDPointing.cpp* and *IOHIDSystem/IOHIDSystem.cpp*. From the archived versions of this project, it seems that the current pointer acceleration mechanisms first appeared in OS X 10.2, released in 2002. However, although the source code is available, the design rationales and principles of operation of these mechanisms are unknown. Figure 2.8 shows the related configuration interface, located in the *Mouse* pane of the system preferences. From a user-perspective, the

³<https://opensource.apple.com/source/IOHIDFamily/>

acceleration mechanisms are badly documented and the tooltip associated to the slider are misleading.

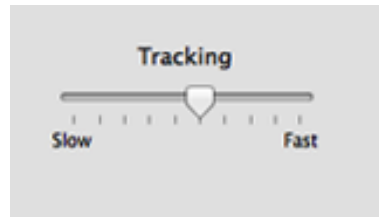


Figure 2.8: OS X 10.6.7 configuration interface for the mouse. A tooltip associated to the slider says *Drag to adjust how fast you want the pointer to follow the movement of your mouse.*

The code from *IOHIDFamily* responsible for pointer acceleration (mainly in *IOHI-Pointing.cpp*) are similar to Windows. Each pointing device has an associated acceleration table provided by its driver (some also define a separate table for scrolling). This table specifies one or more curves defined by a series of segments and a scale level. The slider is shown in figure 2.8 allows users to specify a desired scale among the following:

0(<i>slow</i>)	0.125	0.3125	0.5	0.6875(<i>default</i>)	0.875	1.0	1.5	2.0	3.0(<i>fast</i>)
------------------	-------	--------	-----	--------------------------	-------	-----	-----	-----	--------------------

The system interpolates between the curves provided by the driver to create one that matches the desired scale and maps a vector magnitude to a CD gain value. Then for each (dx, dy) displacement, it computes an approximation of the vector magnitude using the same equation as Windows, uses the created curve to find the right CD gain, applies it to (dx, dy) , adds the previous remainders and returns the integral part of the result after updating the remainders [6].

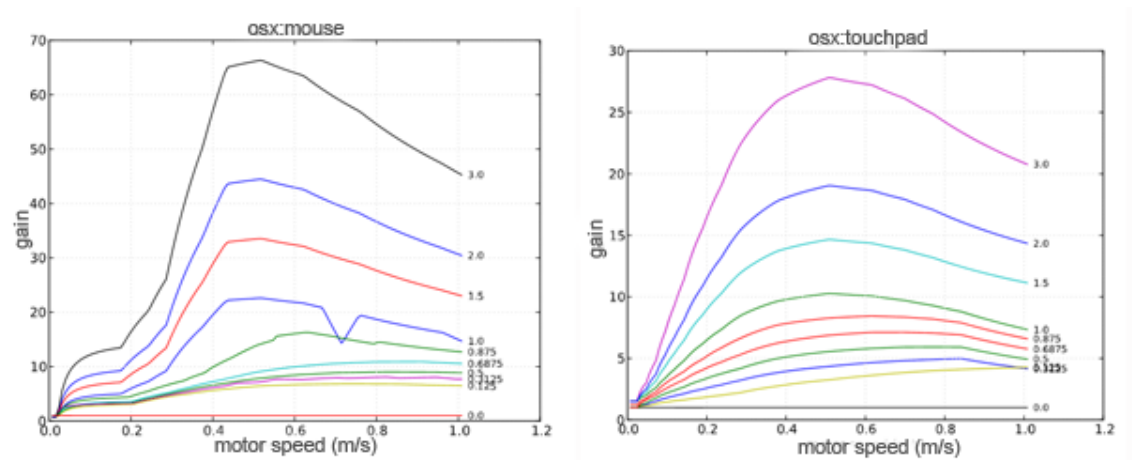


Figure 2.9: OS X 10.6.7 functions for mice (available through the interface shown in Figure 2.8) and touchpads (from [6]).

The acceleration curves stored in device drivers are hardware-independent. When it interpolates between them, the system takes the resolution of the input device into account. However, it uses hardwired constants for the resolution of the display (96 PPI) and the frequency of the input and output devices (67 Hz). A detailed inspection of the drivers available on OS X 10.6.7 showed that several of them use the same tables. There are two particular tables, one for mice and the other for touchpads, that seem to be used by all such devices that rely on Apple's drivers[6]. Figure 2.9 shows the curves of motor speed vs CD-gain associated to each slider position shows in figure 2.8 for generic mice(*left*) and touchpads(*right*) [6].

2.7 libpointig toolkit and EchoMouse

2.7.1 EchoMouse

Casiez and Roussel [6] presented EchoMouse, a device to characterize the transfer functions of any system. It is an electronic device that measures a system's response to pointing movements received from a HID equipment. It investigates the transfer functions used by systems without spending too much time on their internal mechanism. They used it to send data in pointing space, then the response in display space is observed. Repeating this procedure for sufficient numbers of points and conditions to obtain the whole transfer function [6].

EchoMouse constructed by few devices like Microchip PIC (18LF14K50), STM32, Arduino Leonardo. The simplest Echomouse can be built with Arduino Leonardo. In the case of Arduino, EchoMouse receives the number of counts to move by serial port and sends it by HID (as a simple mouse).

The main algorithm used to do that:

- Write mouse and display hardware settings (Resolution and update rate)
- For each available configuration of the mouse speed or acceleration in the system:
 - Perform the following for $N_{points} = 1$ to 127:
 - * Move cursor position to the position $(0, disp_{height}/2)$
 - * Receive the N_{points} by a serial port.
 - * Repeat this until cursor hit the border of the display (remember N_{times}):
 - Send this number by HID Mouse (usually only horizontal direction, since the transfer function is the same in both directions).
 - * Measure the N_{pixels} / N_{times} corresponding to the N_{points} .
 - Measure the N_{pixels} / N_{times} corresponding to the N_{points} .

- At the end achieve full acceleration profile and plot the transfer function [6].

For each available configuration of the mouse speed or acceleration in the system:

- Subpixel processing (how the remainders are handled in the system).
- Correlation between dx and dy. Since, the transfer functions are computed for a single direction, when there are 2D motions, the gain of the transfer function may take into account only the biggest, or a combination of the two. In many cases, the gain is applied to the norm of the displacements ($d = \sqrt{(dx^2 + dy^2)}$).
- Rounding of the gains [6]

2.7.2 libpointig

libpointing is a toolkit developed by Casiez and Roussel to replicate and compare the transfer function used by different operating systems. The goals of libpointing are

- Directly accessing HID pointing devices to bypass the system's transfer functions.
- Replicate transfer functions of all different operating system like Windows, OS X and Xorg.
- Run on different operating system's platform and compare implemented transfer functions with the genuine ones.
- Support comparisons between the replicated functions and other ones.

libpointing supports the use of URIs [22] to specify input and pointing devices, display devices and transfer functions. The main features of the libpointing toolkit are summarized below

Pointing devices

PointingDevice instances are created from URIs using the static create method of that class. Other methods allow checking whether a device is active, to obtain its resolution (in counts per inch), update frequency and URI, and to associate a callback to it. The callback will be executed every time the device has a motion or button event to report, passing it a timestamp, dx and dy values (in counts) and an integer coding the buttons states.

Display devices

DisplayDevice instances are also created from URIs using a static create method. Other methods allow obtaining the horizontal and vertical bounds (in pixels), sizes (in inches or millimeters) and resolutions (in pixels per inch) as well as the refresh rate and the URI of a particular display.

Transfer function

TransferFunction instances are created using a static create method from a URI, a *PointingDevice* and a *DisplayDevice*. Other methods allow to obtain the URI of a function, to clear its internal state and to apply it to dx_{in} and dy_{in} values (in counts) with a *timestamp* to produce dx_{out} and dy_{out} values (in pixels). The toolkit provides subclasses that correspond to different transfer functions. Care has been taken so that all implementations are platform-independent.

Utilities

libpointing includes some test and debugging programs that allow listing the available devices and their characteristics, for example. The toolkit also includes a transfer function plotting tool written in Python using *matplotlib*. This tool proved quite useful as it provided visual confirmation of the implemented transfer function of the Windows, OS X, and Xorg functions matched the data collected using *EchoMouse*.

The toolkit also includes an application that allows testing an arbitrary number of transfer functions at the same time specified by their URI as command-line arguments. The program creates an on-screen cursor (a small square) for each function, a single pointing device being used to control all of them [6].

2.8 Summary

In this chapter, we have described details description about the pointing devices and fundamental terms which help to understand how transfer function works. Later we have discussed most used transfer functions of modern operating systems. As our plan was to work on windows environment so we tried to retrieve the windows transfer function and the inside mechanisms. We have to understand the windows transfer function to deactivate the windows pointing function while our application will be running. This chapter also talks about other transfer functions as the planned application tools going to access them. The plan is to develop a pointing device data logger which will record all the information regarding pointer and pointing devices and an application to allow the user to define their own transfer function. By the studies of the prior work of Casiez & Roussel, we have gathered knowledge about *libpointing* and *EchoMouse*. After being familiar with the *libpointing* repository and *EchoMouse* device, our work becomes easier to deploy the basic concept about how to replicate modern transfer functions.

3 | Concept of User-defined Transfer Function

3.1 Introduction

Buxton stated “one of the things that I see most neglected is any consideration of when to use relative vs absolute control and varying, including when and how to effectively and dynamically switch from one to the other, and when and how to dynamically adjust CD ratio [4].”

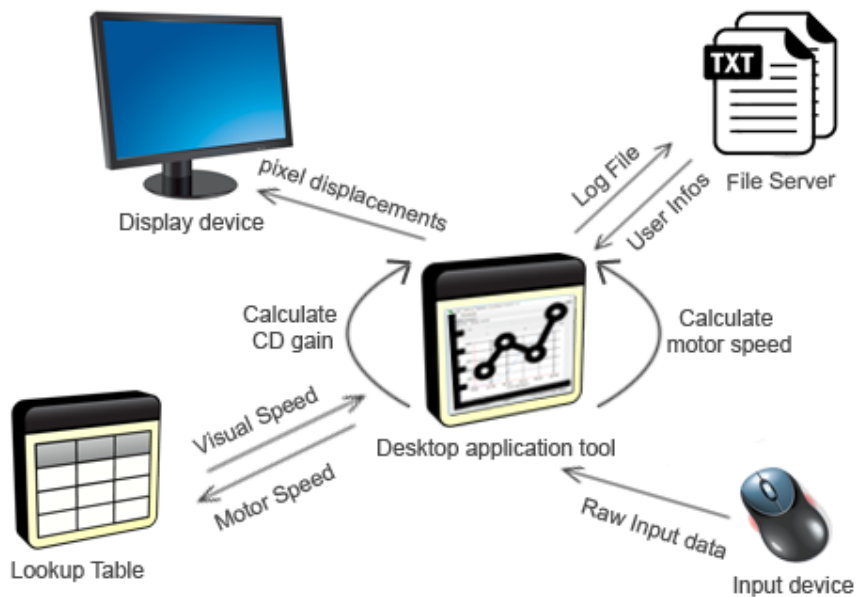


Figure 3.1: Application Process Diagram.

Pointer acceleration is the default transfer function of Microsoft Windows and Apple macOS operating system. It dynamically changes the control CD gain between actual velocity of the pointing device to the actual velocity of the pointer on the screen. An algorithm calculates the transferred pointer data of the transfer function. The physical meaning of the function is achieved by converting the arbitrary values to physical units.

3. Concept of User-defined Transfer Function

A good idea to enact a transfer function is to use a device, which is not attached to the system pointer mechanism and an API or software that provides access to its raw data. As an example, Blanch et al. [2] used the absolute coordinates of a Puck on a Wacom tablet as input for their Semantic pointing techniques.

The figure 3.1 is the process diagram of the planned desktop application tool. The basic plan is to collect the pointing device raw input data(i.e. mickey) and then convert the arbitrary transferred raw data to meaningful physical data. After that, the application has to compute motor velocity (speed of the physical device motion or finger dragging effects) from the physical movement raw data. There is a lookup table for the popular operating system transfer functions. The table contains different ranges of visual velocity (pointer speed on the display) values according to a range of motor velocity values. The corresponding visual velocity is selected from the table and then calculated the CD-gain. The calculated CD-gain is used to compute the visual displacements. Finally, visual displacements are converted to pixel displacements on the screen.

For the user-defined approach, there is an interactive graph section, where users can drag the transfer function curve by the mouse dragging option and move the knot up and down on the curve to define their own transfer function. The manipulated data points are converted to visual velocity data and from the visual velocity data, CD-gain has been calculated. At the end, visual displacements are converted to pixel displacements on the screen.

3.2 Mouse Pointer Recorder

The Idea is to develop a desktop application tool which will be run in the background and will record all the pointing device activities. The application will collect other relevant information from the system to understand an optimal transfer function. Then the application will write all the information immediately into the log file. The application will start each time automatically to collect all the necessary information without interfering the regular work. A separate setting option will be available for the users to change the file destination.

The plan is to retrieve all the raw information of the input pointing devices as well as to collect all the necessary information of the display devices in order to understand a pointing transfer function of any operating system. To uniquely identify each input stream, *timestamp* has been used for each input line. The application tool will write each unique input stream in the text file immediately. It will generate the new log file when the user starts it for the first time. Each time when the application starts it will create a new log file.

To collect the raw input information from the pointing devices, first of all, the application will ensure that the input data are from pointing devices. The type of the devices depends on the position data such as absolute and relative. For example, mouse provides the relative position and touchpad provides the absolute position.

Application tool also collects the input devices button's information and button's raw data whenever a user clicks on the devices. The application will retrieve the data only when the user moved or touched the devices.

The mouse movement provides the displacements values called *Last Mouse Raw Input* on the operating range XY coordinates. When the user moves the physical mouse on the mouse pad (operating range), the system will record displacements values and also the screen cursor position at the same time. The pointer position is an arbitrary pixel value depended on the display resolution and the control devices. The input device position unit is called mickey depend on the pointing device resolution, called DPI(dot per Inch), actually CPI(count per Inch). These arbitrary values are converted to the physical value by dividing screen update rate and system BUS update rate respectively.

All the retrievable information from the pointing devices are listed as *Raw Mouse Flags*, *Button Flags*, *Button Data*, *Raw Buttons* and *Last position of device* on the XY coordinates operating range. All others retrievable information from the operating system are *Screen Resolution*, *Screen Boundary*, foreground running applications and programs information.

3.3 User-defined Application Tool

The simplest way to define a transfer function is to directly map the pointing input device raw values to the output display pointer(cursor) values. In this case, it will be the mouse movement raw input displacement values (x,y) and the display pointer(cursor) position values (x,y) on the system display. The problem is that mouse raw input displacements values are arbitrary values with different units compare to pointer displacements values on the screen.

After collecting the raw data, the first step will convert the both units to similar stander units. For example, the control space and display space collected data units length will be meter and the time will be same units (millisecond or nanosecond). Now if we map physical device movement to the cursor movement then it will be *one-to-one* mapping. It means if the mouse moves one centimeter on the operating range then the cursor has to move one centimeter on the display and the CD-gain will be one.

It is known that all the popular operating system use non-linear continuous transfer function as their sole pointing facilitations. The CD gain is changing accordingly to the change of input device's velocity. It means CD gain will increase when the device moves faster and CD gain will decrease when the device moves slower.

3.3.1 Defaults Transfer Function

To define the existing operating system's transfer functions, there will be a graphical user interface where the user can select a function from the selection box and after that apply the selected transfer function to the system cursor. For better presentation, there will be a plot section to plot the selected transfer function.

The plan is to use the transfer function data points from the *libpointing toolkit*, which are generated by the *EchoMouse* pointing device developed by Casiez and Roussel [6]. In their published research named '*No more Bricolage! Methods and Tools to Characterize, Replicate and Compare Pointing Transfer Functions*', they used *EchoMouse* pointing device and *libpointing toolkit* to replicate all the popular operating systems transfer functions such as Windows, Linux, macOS etc.

There will be file structure system at the application backend, where all the transfer functions data points will be stored from *libpointing* repositories and organized by the name of the different operating system and their default setting. The application will load particular transfer function file from the corresponding data folder, according to the selected transfer function name and pointer speed setting.

the total number of selected transfer functions will be highly dependents on the collected data point sets of the *libpointing toolkit*.

3.3.2 Interactive Graph

The main objective of this thesis is to implement an application tool, which will allow the user to define their own transfer function. To fulfill this objective we have planned to introduce a technique to manipulate a range of data points in a convenient way. Our plan to use interactive graph system to manipulate the data points by the user interaction with the graph. In the interactive graph, the user can change the plotted curve by dragging the knot up and down on the curve with the help of mouse dragging option. By moving the knot from one place to another the line will move, the curve of the graph will change and data will be manipulated. After that, the application will use the customized data to define new transfer function and by this way, the user can define their own transfer function.

There will be saved and load option for the user-defined transfer functions. The user could save the manipulated data sets into a text file and also could load it for applying to the system pointer. There will be a feature to apply the newly defined transfer function directly to the cursor.

3.4 Summary

The collected data points from the *libpointing toolkit* make the work easier to replicate existing transfer functions and interactive plot graph idea helps to define user's

own transfer function. Windows operating system also uses predefined transfer function curve with 5 inflection points in the lookup table and others points linearly extrapolated beyond the last inflection point. Similarly, the planned application tool will use collected data points from the prior research as a lookup table. The interactive graph curve system will have 6 knots, (the first pair will be [0,0] like windows) which will represent 6 data points on the transfer function curve. Therefore, the application will use spline interpolation to convert those 6 data points to 127 data points. Because of each replicated transfer function from the *libpointing toolkit* has 127 data points. The HID specification defines a simple boot report format for the pointing devices mice and that format uses only one byte per axis [3]. It represents -127 to +127. The transfer function data points stored in the lookup table(text file) are absolute values of motor speed upon the corresponding visual speed from the range 0 to 127. The Appendix table A.1 shows a set of data points for the windows transfer function when pointer speed is 6 and the enhance pointer precession is checked.

4 | Design and Implementation

4.1 Introduction

To implement our basic concept, first of all, we try to retrieve the raw input values from the pointing devices and then write the retrieved data into a log file. For the purpose of data retrieve and store into the file, we developed a desktop application tool called *Mouse Logger*. We have developed another separate application called *User-defined Transfer Function Tool* to allow the user to define their own transfer function. In the *User-defined Transfer Function Tool*, the user can select different types of transfer functions of the recent operating systems and also can customize their own transfer function. This application tool is the main implementation part of this thesis and at the end, some features are added to collect user's devices information and feedbacks to conduct a user study. This chapter includes separate details description of the both desktop application tools.

4.2 Mouse Logger

Mouse logger is a universal mouse recorder, it records mouse movement, mouse raw data and pointer position on the screen. The aim is to collect as many information as possible to understand the pointing transfer function of the operating system and the whole pointing facilitation mechanism. The application will run in the background without any interface and all the setting can be changed from the system tray icon.

4.2.1 Architecture and Requirements

Figure 4.1 is about the simple architecture of the Mouse Logger application. It shows that application collects data from the input devices and write it into a text file. The data flows in one direction. The objective of the application is very simple, it collects information about the input-output devices and stores it into the log file. Each time when the application starts, it creates a new log file in the user's documents folder. The created file name contains current *timestamp*. Every time

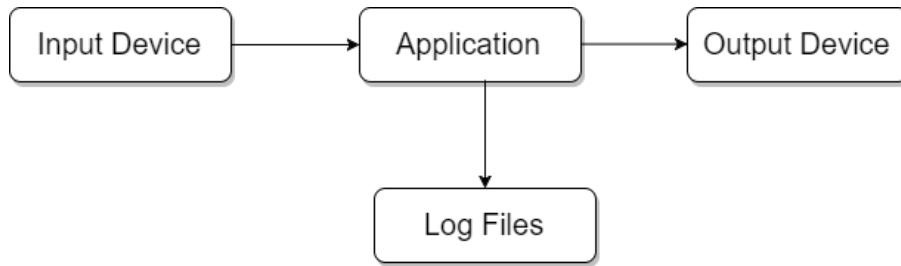


Figure 4.1: The simple architecture of the MouseLogger Tool.

when a new file is created, the application always checks the existence of the old files. The backup process of the application searches the old file and moves it into the backup folder. The application registers itself in the system startup list when it starts for the first time. By default application will write the log files in text format but the user can change the file format into comma-separated values (CSV).

This application developed on *.Net(dot net) Framework 4.5.2* so it will run only in Windows environment and minimum version requires Windows 8.

4.2.2 Design

The Graphical User Interface (GUI) (figure 4.2) mainly contains the file location path where the log file is saved. By default, it creates a log file in the user's documents folder but the user can change the file destination.

The application starts with a notification message (figure 4.3 (left)) showing that the application is running and the setting could be changed from the system tray icon. The application system tray icon has right mouse click option, a strip menu will appear after the click with the options (figure 4.3(right)) pause, *change file path* and close the application. The application interface will appear on the screen by the click of the *Change Text Path* menu item or double click on the application system tray icon. User has to pause the application before changing any setting. After clicking on the *Pause* button, the user can change the file destination by creating a new file. The application will write log data into the newly created file.

The user can choose the file format from text or CSV, by default the log data will write in text file but there is a *checkbox* option labeled CSV. If the *checkbox* option is checked then the file will be written in CSV format. The application registers itself in the system startup program list when it is started for the first time. There is a *Clear Apps from Startup* button in the bottom left corner of the application for removing the application from the startup program list. At the center of the application, there is a big multiline text box as a status box, whenever changes are made by the application it shows as a feedback. Consequently, user can check the status during the application running time. At the upper right corner, there is a *Hide* button to hide the whole application and run it in the background.

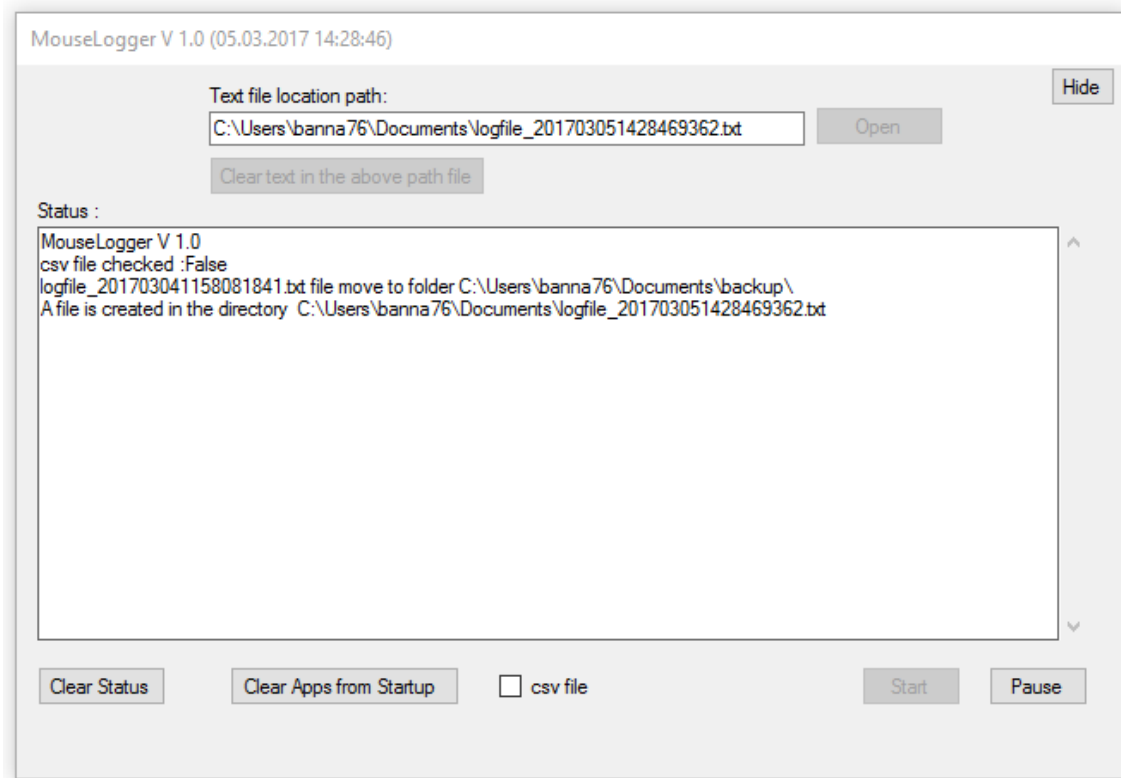


Figure 4.2: The screenshot of the Mouse Recorder Application Tool.

4.2.3 Implementation

The class *Form.cs* contains all the graphical code, *Win32 APIs*¹ function declarations, *User32.dll*² library functions, mouse events and device information functions.

The *Form.cs* class used *Win32 APIs* to collect display devices and cursor information data. There is a message filter interface to declare the global mouse events class. All types of mouse events are defined in the mouse event class. There are also extended mouse events classes to provide additional information about mouse click and movement actions. The application mainly used mouse events for the input pointer devices movement information and click events for taking data whenever the user clicked on the mouse buttons. These global mouse events help to record the cursor position information whenever the application run in the foreground as well as the background.

There is a *PreFilterMessage* function in the *Form.cs* class and this function execute each time when the input pointing devices are moved or touched. *PreFilterMessage* function always checks the pointing devices ID(Identification number) and the type of the input devices. The application starts taking raw input data from the pointing

¹[https://msdn.microsoft.com/en-us/library/windows/desktop/ff818516\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/ff818516(v=vs.85).aspx).

²<http://www.pinvoke.net/default.aspx/user32.getwindowrect>

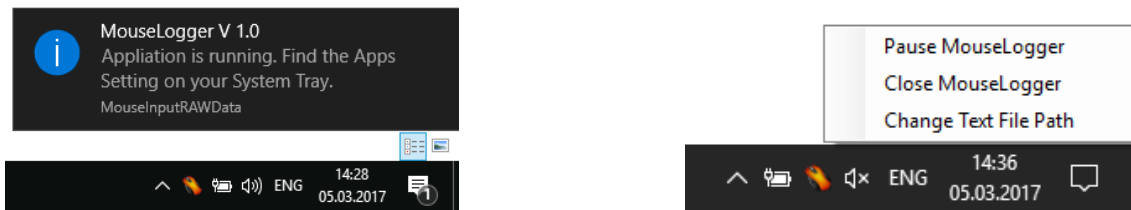


Figure 4.3: The pop-up notification message (Left) and System Tray Icon right click menu options (Right).

devices only when the input device type match. The application starts polling information every time when any events occur such as device move, button click, wheel scroll etc.

The class named *MouseInputRAWData*, where all the raw information are declared and a method are also defined. This class returns the device IDs and input device type. When the input device is moved or touched it checks the device type, if it is mouse then returns a string value *mouse* after that application start polling pointer position values (x,y) by the global mouse events. The application also collects data like the name of the currently running foreground program, virtual screen width, virtual screen height.

At the same time *PreFilterMessage* method collects the pointing device's *last raw values* in x direction and *last raw values* in y direction on the operating range. The operating range of the device is the maximum of unconstrained physical movement area. It is the displacement of the physical mouse movements or the finger dragging effect on the touchpad. Application collects mouse extra information, which are additional information for the mouse events. Mouse *ButtonFlags* is the flag for the event. Raw *MouseFlags* indicate the state. *MouseFlags* value 0 means *MoveRelative*; Relative to the last position, *MouseFlags* value 1 means *MoveAbsolute*; Absolute positioning, *MouseFlags* value 2 means *VitualDesktop*; coordinate data is mapping to a virtual desktop, *MouseFlags* value 4 means *AttributesChanged*; Attributes for the mouse have changed. *ButtonData* actually contains the delta amount, when the mouse is moved. *RawButtons*, contains raw button data.

The application also collects some other calculated values such as mouse vector distance, cursor vector distance, velocity of the pointing device and velocity of the cursor. After that, the application calculates CD-Gain for each collected raw input value of the pointing device for each unique *Timestamp*.

4.2.4 Log File

The first attribute of the log file is (see figure A.1) *Timestamp*, the format of the Timestamp is year-month-date-hour-minute-second-millisecond. The second attribute is the *DateTime* ticks, a single tick represents one hundred nanoseconds or

one ten-millionth of a second. There are 10,000 ticks in a millisecond or 10 million ticks in a second. After that x/y values of the pointer on display screen are there, next value is the foreground application name and then the virtual resolution of the current system display. This virtual resolution is the height and width of the virtual screen so if the current system has multiple extended displays then it will show the total height and width of the system screen. Next value is a point decimal number, which is the vector distance from the previous cursor position to the current position. Next attribute is the last raw mouse motion values x/y in the direction of x -axis and y -axis respectively. Other values are mouse additional information, mouse button flags, mouse button raw data, mouse flags, mouse raw button values, next decimal values indicates the distance of previous raw mouse motion to current mouse motion. The very last value is the CD-Gain for each unique *timestamp*.

4.3 User-defined Transfer Function Tool

A user-driven approach used to assess the system's *transfer functions* based on natural interaction. We developed a desktop application tool for the users to choose a specific *transfer function* from the selection list and select different pointer speed. Afterward, users can apply that function to the system cursor. The user also can declare their own *transfer function* by the help of the interactive transfer function curve.

4.3.1 Architecture and Requirements

Figure 4.4 shows the simple architecture of the user-defined application, which is similar to the mouse logger application.

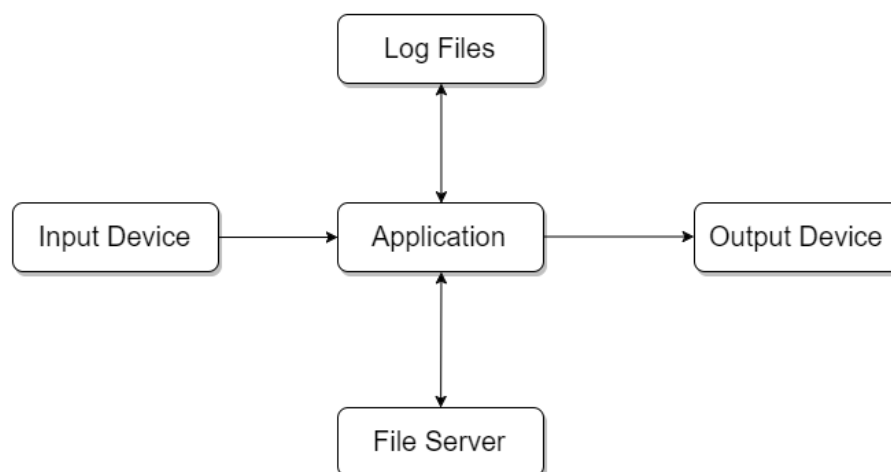


Figure 4.4: The simple Architecture of the User-defined Transfer Functions Tool.

The desktop application tool collects raw input data from the input devices and

4. Design and Implementation

calculates the physical mouse motion velocity from the raw data. After that, It retrieves the corresponding visual velocity of the pointer from the stored data points sets. At the end calculate the CD-Gain and replace the current cursor position according to the CD-Gain. The application also has interactive graph curve option, where a user can customize all the existing operating system's transfer functions and defines their own new transfer function.

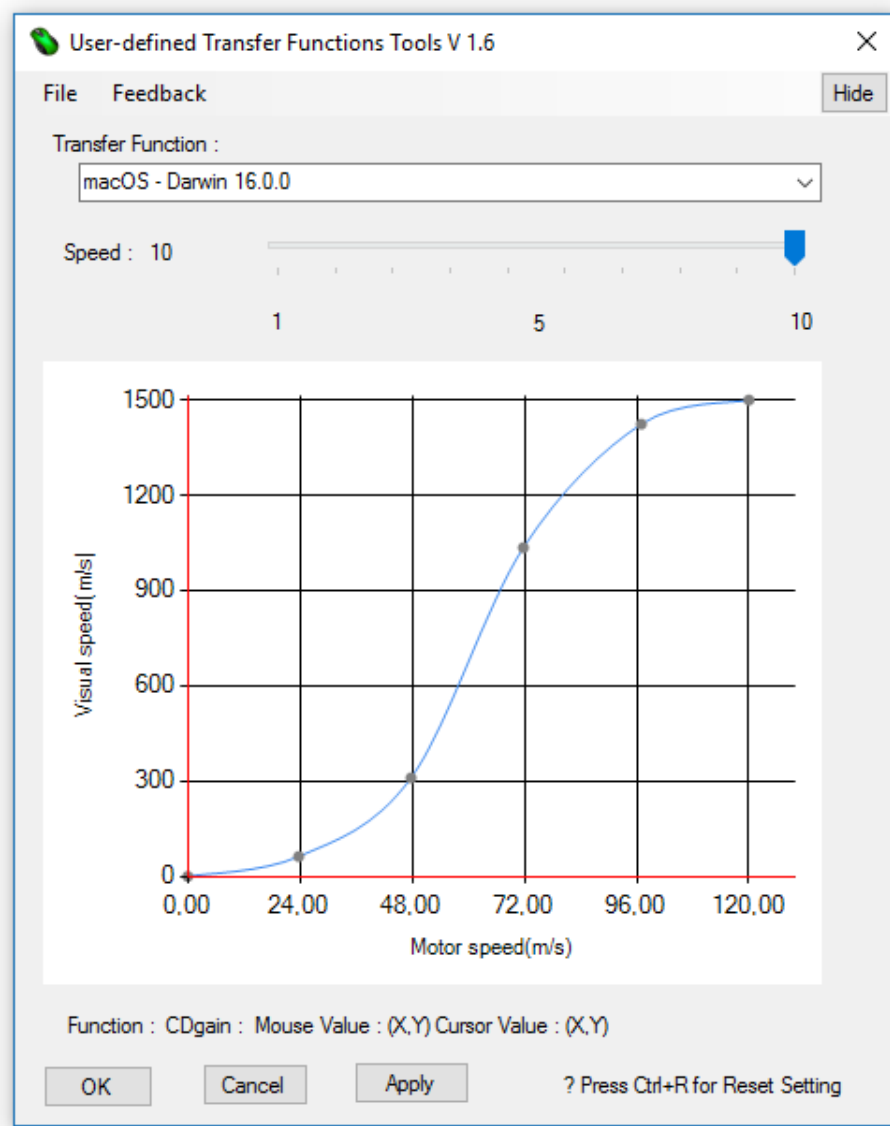


Figure 4.5: The screenshot of the User-defined Transfer Function tool main interface.

The application requirements are same as mouse logger application tool. It developed on *.Net framework 4.5.2*, and used *Win32 APIs* for access *User32.dll* library functions. In addition, the application also used *Google Sheets APIs*³ to collect user device's information and feedbacks remotely from the user machine for user study purpose.

³<https://developers.google.com/sheets/api/guides/concepts>

4.3.2 Design

The application starts with a notification message (see figure 4.8(left)) that the application setting could be changed from the system tray icon. The system tray icon has right mouse click options (see figure 4.8(right)). The right mouse click strip menu contains three options, first one, application view option, where a user can view the whole application interface. The second option, apply the currently selected transfer function to the pointer and third application terminate.

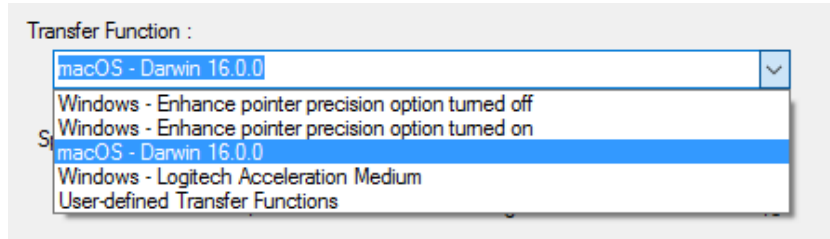


Figure 4.6: The screenshot of the selection combo box.

The application main interface (see figure 4.5) contains a file menu, which has open, save and reset system's transfer function options. It has a selection box(see figure 4.6), where a user can select a transfer function from the list. The application retrieves the folder name from the selected transfer function name. It also has a speed trackbar according to the default operating system's pointer setting. It takes the value of the trackbar whenever the user changes the value. After that combining this trackbar value with the folder name, the application generates a file destination path and searches data points in the stored file. If the application successfully collects data point set from the generated file then it plots the transfer function in the interactive graph plot section. Now user can apply this transfer function to the system's cursor by simply clicking on the *Apply* or *Ok* buttons.

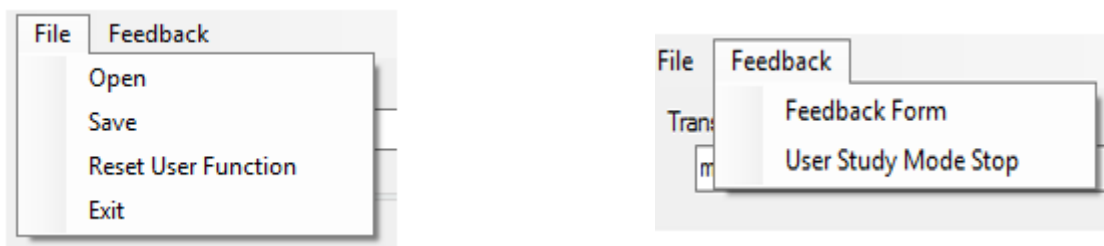


Figure 4.7: The Application main menu (Left: File Save and Open Option, Right: Feedback Option).

In the interactive graph option, the user can simply move the knot up and down on the curve to move the curve position and generate a new transfer function for the user-defined option. There are six knots on the curve and the user can hold any one at a time to move up or down direction by the mouse dragging option. After that, the modified selected transfer function could be directly applied to the system's cursor by simply clicking on the *apply* or *ok* buttons. If the user clicks on

the *ok* button then the transfer function will be applied to the system cursor and the whole application will be invisible.

The application also has saved and load options. User can save their customized transfer function curve from *File > Save* option(see figure 4.7(left)). When a user clicks on the *Save* menu item, the application will request the user to create a file at their desired location. After that, the modified data points will be saved under the filename at the selected destination. The user can also load the saved file from *File > Open* option. By this feature, the user can save their favorite transfer functions and move it from one machine to another.

There is a feedback item (see figure 4.7(right)) in the main menu to show feedback dialog box and activate/deactivate user study mode. The *cancel* button action will remove all the applied transfer function setting from the cursor and reset the cursor to default operating system setting. If the user applied any worst mapping transfer function to the pointer which is so fast or too slow for operating the pointer. For that, there is a hot shortcut key to reset the application to the default operating system setting by pressing **Ctrl+R** keys on the keyboard.

4.3.3 Implementation

4.3.3.1 Notification and System Tray Icon

To implement the notification (see figure 4.8(left)) feature we have used *Tulpep Notification Popup Window*⁴ package downloaded from NuGet Gallery.



Figure 4.8: Start notification message (Left) and System Tray Icon right click menu options (Right).

4.3.3.2 Load libpointing Data Points

In the research work of Casiez and Roussel [6], they presented *EchoMouse* a device to characterize the *transfer functions* of any system, and *libpointing* a toolkit to replicate and compare the *transfer functions* used by Windows, OS X and Xorg. The application takes all the *transfer functions* data points sets and stores them in a file structure system (see figure 4.9).

⁴<https://www.nuget.org/packages/Tulpep.NotificationWindow/>

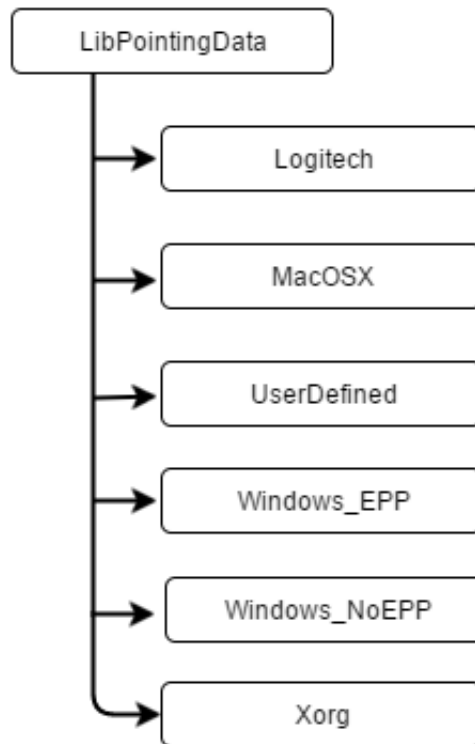


Figure 4.9: File structure system of the libpointing data points.

The application loads data points from the file when the user selects a particular transfer function and also plots the curve in the interactive graph section. Each default *transfer function* has a folder by its own name and inside the folder, there are files according to their default pointer setting. For example, Windows operating system has two folders named *Windows-EPP* and *Windows-NoEPP*. According to Windows mouse properties panel, *Windows-EPP* abbreviates Windows enhanced pointer precision. There are eleven files inside the *Windows-EPP* folder named from *f1* to *f11* for representing all the different *transfer functions* of Windows at different pointer speed setting when *enhanced pointer precision* is enable.

Data points are represented as comma separation values of 127 rows and 2 columns inside each function file. First column of integer values represents the *motor speed* (also known as *physical mouse velocity*) and second column decimal fraction values represents the *cursor speed* (also named as *virtual pointer velocity*). Each *transfer function* file contains 127 different pairs of values which are *motor speed* and corresponding *visual speed*. The Table A.1 shows full set of data points of Windows transfer function when pointer speed is 6 and enhance pointer precision is checked on.

4.3.3.3 Inside Calculation

The main algorithm of the user-defined transfer function tool listed below :

4. Design and Implementation

- When the user clicks on the *Apply* button, the application retrieves the input device type.
- If the input device is pointing device then it starts polling mouse X and Y values in a time interval.
- For each different X and Y values, the application calculates the norm of the displacements $d = \sqrt{dx^2 + dy^2}$
- This norm of the displacements consider as mouse speed
- Now if the user customized interactive transfer function curve then application calculates pointer speed from the modified curve data points else the application retrieves pointer speed from the stored file according to the mouse speed.
- Calculates the $CD_{gain} = Speed_{pointer}/Speed_{device}$
- Retrieves the pointer current position and applied the CD_{gain} to calculate new position
- Verify and convert new position values to display pixel values (x,y)
- Set the calculated pixel values to the system pointer
- Go back to the first step

When user moves(mouse) or touches(*touchpad*) the pointing devices, the application polling raw input values from the input devices. After that, it checks whether the input data coming from pointing device by checking the device types. If input data are coming from pointing devices then application starts recording input raw data in a millisecond of the time interval. The application calculates the motor speed for each input values from the *last input raw* value. The *last input raw* value is the displacement of the input device on the operating range. The norm of the displacements is d ,

$$d = \sqrt{dx^2 + dy^2} \quad (4.1)$$

The method *GetVisualSpeed* with motor speed parameter calculates the visual speed values from the motor speed. The *motor speed* are declared as an integer in the stored data points file but calculated norm of the displacements values are always fractional decimal values. For this reason, the application takes two integer values x , the floor value of the motor speed and $x + 1$, the ceiling value of motor speed. Now the visual speed V_{speed1} and V_{speed2} have been retrieved from the stored data points sets respectively for two motor speeds. If V_{speed1} is equal to V_{speed2} then return V_{speed1} as visual speed or else application calculates the visual speed by the following equation

$$V_{speed} = (1 - fractionsValue) * V_{speed1} + fractionsValue * V_{speed2}; \quad (4.2)$$

Here *fractionsValue* is the fractional part of the decimal motor speed value.

If the user moves the interactive graph curve, the application calls *GetVisualSpeedFromGraph* method with motor speed parameter. When the user moves the knot the transfer function graph curve bent according to the knot movement by the help of curve spline interpolation function and the data points of the graph has been changed. The interactive graph provides 6 knots which represent 6 pairs of data points on the curve. Therefore, the application executes *getDataPointsFromGraph* method with user chart parameter. This method takes the position of 6 knots as new data points and calculates 127 data points by the help of our *Spline Interpolation* class. The *Interpolation* class generates 121 unknown data points from this 6 known data points. The *GetVisualSpeedFromGraph* function calculates the visual speed from the motor speed when all the 127 data points are available. This calculation is same like previous technique and the method returns visual speed as a function return value.

The application calculates the CD_{gain} when it gets visual speed for the corresponding motor speed. To calculates the visual speed of the cursor on the X and Y direction we have used following equations

$$x_{visual-speed} = Raw.Input.LastX * CDgain; \quad (4.3)$$

$$y_{visual-speed} = Raw.Input.LastY * CDgain; \quad (4.4)$$

When the visual speed in X and Y direction are calculated, the application converts those speed values to pixel points on the screen by the following equations

$$x_0 = System.Windows.Forms.Cursor.Position.X + x_{visual-speed}; \quad (4.5)$$

$$y_0 = System.Windows.Forms.Cursor.Position.Y + y_{visual-speed}; \quad (4.6)$$

Therefore, the current X and Y position of the cursor on the screen have been taken and then added visual speed with them. The application considers this visual speed as the calculated visual displacements of the cursor on the screen. This calculated (x_0, y_0) values have been checked whether the values are in between the virtual screen boundary or not. If the values are in the display boundary range then application converts (x_0, y_0) values into an integer for fits as screen pixel values and replaces the current cursor position to the calculated position. If the calculated values exceed the maximum display boundary values then the application takes the maximum display resolution values. In other cases, the current cursor position remains unchanged.

4.3.4 Log File

The user-defined application tool generates four log files in the system's user document folder, two of them for feedback information which we will discuss in the next chapter and others two are application log file. One for recording all the changes made by the user and other is for customized transfer function data points sets. When the application runs for each time it creates a log file (see figure A.2) with current *timestamp* in the name and all the information are written in it. Whenever the user selects any transfer function from the selection box, log file writes the name of the transfer function and after that start writing all other information. The recorded data in the log file start from the timestamps, then inputs raw values of pointing devices in X and Y directions, after that motor speed of the physical motion of the input device, and then the visual speed of the pointer, next value is the calculated CD-Gain, at the end X_0 and Y_0 current pixel position of cursor on the screen.

The figure A.3 shows another log file for the interactive graph system when the user manipulates the curve data points and applies the customized transfer function to the cursor, the application writes whole new customized transfer function data points into the log file. Each time when the user changes the curve and applies it, application appends new transfer function data points sets in the log file immediately.

4.4 Summary

We have developed *Mouser logger* application tool to collect as many information as possible for the pointer and pointing input devices and after that write them into the log file. But there are different input devices with different device identification number and they varied according to the operating systems and input devices types. Different participants machines return different input devices identification numbers. In the initial plan, it was hard to recognized pointing devices by the device ID. After that, we have used input device type to recognize pointing devices. There are also some delay and buffering issues during the log files writing.

The user-defined transfer function tool, where the user can select transfer function from the selection option and customize the function to make their own transfer function is the main implementation part of this thesis. The total number of existing operating system's transfer functions are limited according to the availability of *libpointing* data points sets. There are some other features such as collecting user demographic information, user input-output devices information and remote data transfer from the user's machine to *Google Sheet*. Which we will discuss in the next chapter.

5 | Study for Interesting Transfer Function

5.1 Introduction

In this chapter, we will discuss how the user study is conducted by the developed desktop application tool to identify interesting transfer function. The detail of the user study step by step starting from design, method, result, discussion and so on. The chapter begins with the design deployment of the application and the details description about participants, apparatus, procedure and after that present the result of the study. Finally, analyze the result to identify interesting transfer function for touchpad and mouse.

5.2 Design Deployment

The application has added some additional features for user study purpose. The extended features are notification message box, feedback form, and Google spreadsheet. Google spreadsheet is used as a database to collect user study information.

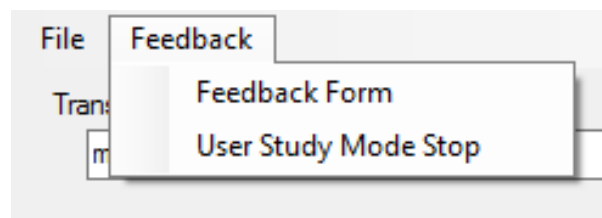


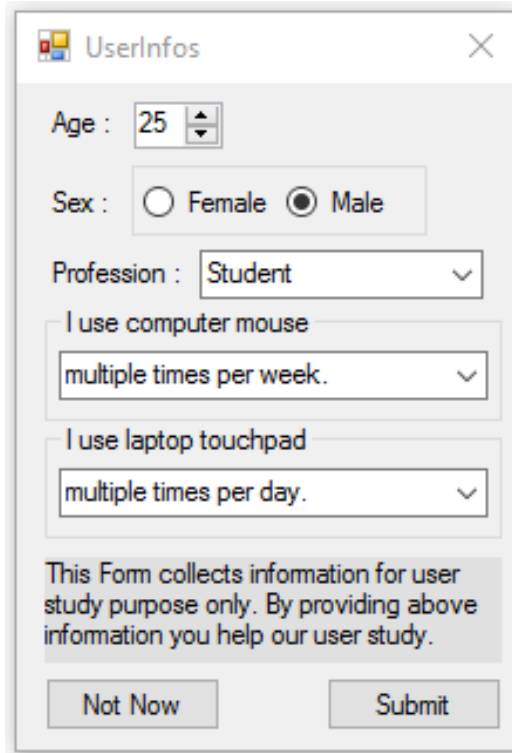
Figure 5.1: Feedback options in the main menu.

The application has a link in the main menu to go to the feedback dialog box and also has an option to activate or deactivate user study mode (figure 5.1). To request the participants to provide feedback in appropriate time with a convenient way, a pop-up notification message will appear in regular interval of time while any transfer function is active. The intention is to get the most effective feedback in

an appropriate time. Initial thought was to only write the feedback log file in the user's document folder, but later idea has been changed and added *Google Sheets* to collect user feedback information remotely from the user's computer.

5.2.1 Extended Graphical User Interface

An information dialog box (figure 5.2) will appear when the participant starts the application for the first time. This dialog box will take demographic information to describe and divide participants into different groups. This form also collects feedback about pointing devices information from the participants. The collected information is age, sex, and profession from the demographic information. The application tries to estimate participants preferable pointing devices by asking the question like how often they used mouse and touchpad.



The image shows a Windows-style dialog box titled "UserInfos". It contains the following fields and controls:

- Age: A spin box with the value "25".
- Sex: Radio buttons for "Female" and "Male", with "Male" selected.
- Profession: A dropdown menu with "Student" selected.
- I use computer mouse: A dropdown menu with "multiple times per week." selected.
- I use laptop touchpad: A dropdown menu with "multiple times per day." selected.
- A text area containing the disclaimer: "This Form collects information for user study purpose only. By providing above information you help our user study."
- Two buttons at the bottom: "Not Now" and "Submit".

Figure 5.2: Demographic information dialog box.

A pop-up notification message (figure 5.3) will appear each 5 minutes interval during application running time. The notification message will only pop-up when participants used any transfer function to the cursor. This notification is a graphical control element that communicates to the participant without forcing them to react to the notification immediately. The notification message shows a question like '*Do you like currently active transfer function?*'. The notifications usually disappear after 20 seconds. If the participants wish to provide feedback they can do it anytime from the *Feedback* link in the main menu.

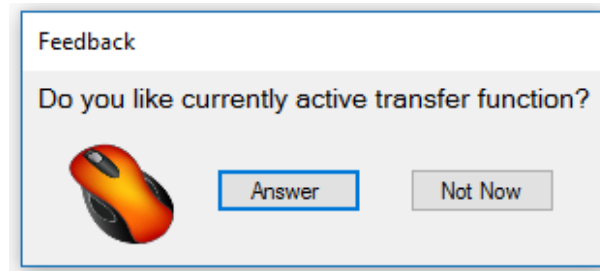


Figure 5.3: Pop-up notification message box reminds the participants for providing feedback about the currently activated transfer function.

If participant clicks on the *Answer* button on the pop-up notification or on the *Feedback Form* link from the main menu a questionnaire dialog box will appear (see figure 5.4). The feedback questionnaire form has six *Likert scale* questions with a choice of answers set.

Each question has a statement and the participants asked to evaluate by giving it a quantitative value of an objective dimension, with a level of agreement or disagreement. The format of our five-level Likert item are:

1. Strongly disagree
2. Disagree
3. Neutral
4. Agree
5. Strongly agree

The 6th question is an optional comment for any kind of opinion regarding the application or transfer function. When the participant clicks on the *Submit* button application submits the feedback to the *Google Sheets* and also write information in the log file on the local machine.

5.2.2 Log File

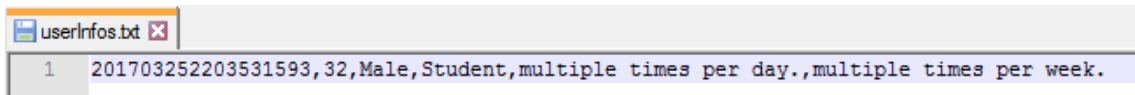
When application starts for the first time, it will create a *userInfos.txt* (see figure 5.5) text file with the user identification number(ID) in the user's documents folder. Which is the unique identification number for the user generated by the combination of timestamps (*yyyyMMddHHmmssffff*). For example, 201703252203531593. This user ID will never be changed until the user deleted the userInfos text file from their local machine. If the user deletes this file from the user's documents folder then the application will again generate this file with the new user ID and consider the participants as a new user. If participant clicks submit button on the *UserInfos* dialog box then this file will write rest of the information from the dialog box information, otherwise, it will write only the user ID. If this file misses all the

The image shows a 'FeedbackForm' dialog box with a close button (X) in the top right corner. It contains five Likert scale questions, each with five radio buttons ranging from 'Strongly disagree' to 'Strongly agree'. The first question, '1. The selected transfer function is too fast.', has the 'Strongly agree' option selected. The second question is '2. The selected transfer function is too slow.', the third is '3. The selected transfer function fits my needs.', the fourth is '4. The selected transfer function is jumpy.', and the fifth is '5. The selected transfer function is smooth.'. Below the questions is a text area for '6. Comments (optional)'. At the bottom, there is a disclaimer: 'This Form will collect System Input & Ouput devices information for study purpose only. By providing the information you help our user study.' and two buttons: 'Cancel' and 'Submit'.

Figure 5.4: Feedback questionnaire dialog box for user study purpose.

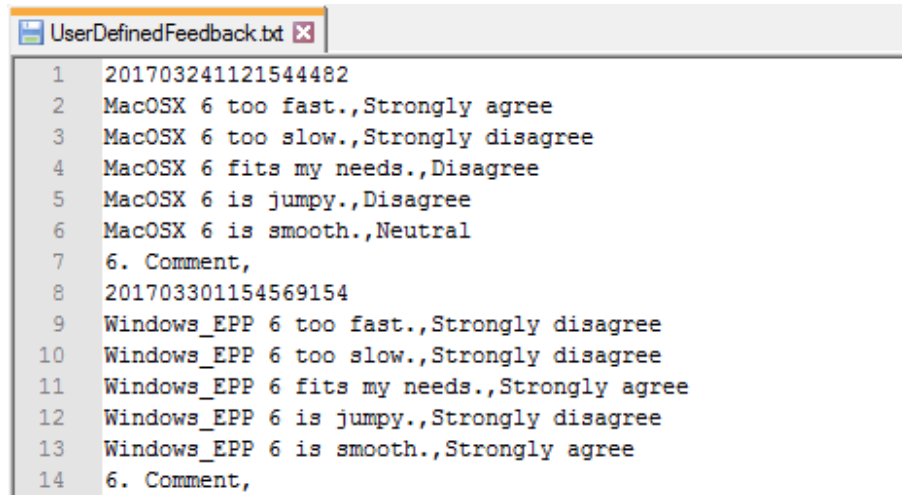
require information then *UserInfos* dialog box will appear each time whenever the application starts.

The application generates another log file (see figure 5.6) when a participant submits any feedback. This log file writes all the questions and answers immediately after submission including timestamps. The question contains sufficient information to identify a specific transfer function, such as the name and speed of the particular transfer function, the corresponding answer is separated by comma separation in each line. If the participant provides more feedback then next set of data will append into the same file.



```
1 201703252203531593,32,Male,Student,multiple times per day.,multiple times per week.
```

Figure 5.5: User Information text file.



```
1 201703241121544482
2 MacOSX 6 too fast.,Strongly agree
3 MacOSX 6 too slow.,Strongly disagree
4 MacOSX 6 fits my needs.,Disagree
5 MacOSX 6 is jumpy.,Disagree
6 MacOSX 6 is smooth.,Neutral
7 6. Comment,
8 201703301154569154
9 Windows_EPP 6 too fast.,Strongly disagree
10 Windows_EPP 6 too slow.,Strongly disagree
11 Windows_EPP 6 fits my needs.,Strongly agree
12 Windows_EPP 6 is jumpy.,Strongly disagree
13 Windows_EPP 6 is smooth.,Strongly agree
14 6. Comment,
```

Figure 5.6: Feedback log file.

5.2.3 Google Sheets

The *Google Sheets* is introduced as data storage for this application. Google spreadsheet is used to collect data remotely from the user's machine. The figure 5.7 shows extended architecture of the application. There are separated two tables for collecting all the necessary and sufficient data for evaluating a transfer function. One table is for feedback and other is for the trail. Figure 5.8 show the Entity-Relationship (ER) model between two table.

The feedback table contains *timestamp* as a primary key, *userID* as a foreign key and other 12 attributes. The application collects user pointing devices information, display devices information, demographic information and feedback question-answer sets. The application also collects each transfer functions activated time for each trial and the whole customized transfer function as a data points set. When a participant submits the feedback by clicking the *Submit* button on the feedback form, all the collected information have been uploaded to the Google spreadsheets. The application never records any kind of data that can individually identify any participants.

The backend data storage contains another table for the trial, which is similar to the previous table with the primary key *timestamp*, foreign key *userID* and other 11 attributes. Almost all of the attributes are same as the previous table except selected transfer function name and speed attributes. Therefore, this table is not for feedback so it does not have feedback question and answer columns. The information is inserted into this table at the end of a trial when a participant clicks on the *Cancel*

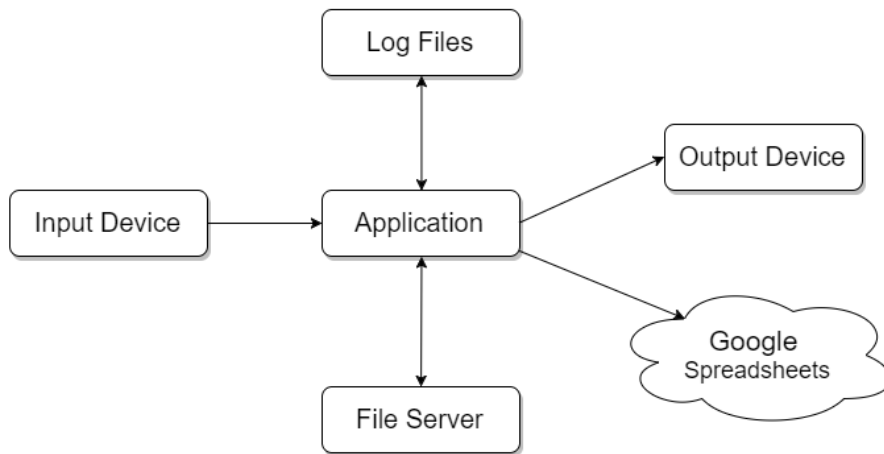


Figure 5.7: The application architecture after adding Google spreadsheet.

button. The aim is to retrieve all the trials information for a particular participant in the certain environment. Later it will help us to analyze the pointing transfer function of the system. The *userID* is the foreign key for the both tables, so a link can be made between this two tables by the foreign key. There could be more than one feedback from one particular and also more than one customized entry in the trial table. The database table entity relation is many to many.

googlesheets.cs and *googlesheetsCustomData.cs* classes contain all the code for implementing this remote data backup features. *Google APIs* and *Google APIs Sheets v4* packages used for this purpose. *Google APIs Core* and *Google APIs Auth* packages also used for authorizing the access of the *Google Sheet*. The *Google Sheet* is identified by the sheet ID. There is a unique sheet ID for each google document. There is an access token json file named *client-secret.js* for accessing the google sheets. The class has a constructor with all the necessary parameter. All the uploading data are passed as a list of array parameter of the constructor and appended them to the *Google Sheets*. Information is appended by *Google API append service request* and values are inserted into the table as like user entered values.

5.3 Participants

In 3 weeks of the study period, in total there were 140 trials and 57 feedbacks. There were 23 unique user entries in the both tables, so a total number of participants was 23. These 23 participants completed at least one trial. So we can state that 23 participants took part in the study there 20 were males and 3 were females. The average age of the participants was 28.83 (SD: 3.16). The participants were mainly students (19 out of 23) with different subjects. The other 3 of them worked as employees in the information technology profession and an academic person.

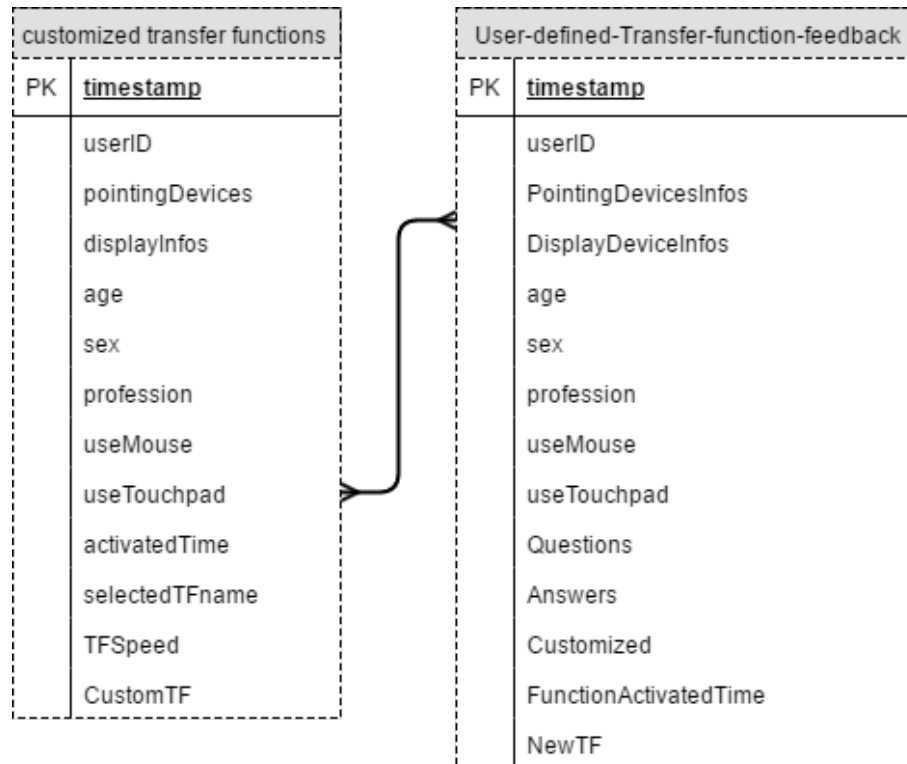


Figure 5.8: GoogleSheets tables and Entity-Relationship (ER) model.

5.4 Apparatus and Materials

Most of the participants preferred to use a mouse as a pointing device, 16 out of 23 participants used a mouse ‘*multiple times per day*’, 5 participants used a mouse ‘*multiple times per week*’ and 2 used a mouse ‘*multiple time per month*’.

For a touchpad pointing device, 12 out of 23 participants mentioned that they used a touchpad ‘*multiple times per day*’, 4 mentioned that they used one ‘*multiple times per week*’, 6 used one ‘*multiple time per month*’ and one participant never used a touchpad.

There were 5 participants were common they stated that they used a touchpad and a mouse multiple times per day. The application retrieved the input pointing devices information during the application running time and found that 4 out of this common 5 participants have both a mouse and a touchpad as a pointing device. The application collects input pointing device IDs and device description information for retrieving the pointing devices. The collected information states that 15 out of 23 participants have *USB Input Device* (which considered as a mouse device) along with a pointing device touchpad. There were 15 participants which have a mouse and a touchpad both pointing devices in their machine during the application using time and rest of the 8 participants only have a touchpad or trackpad as pointing device attached to their machine.

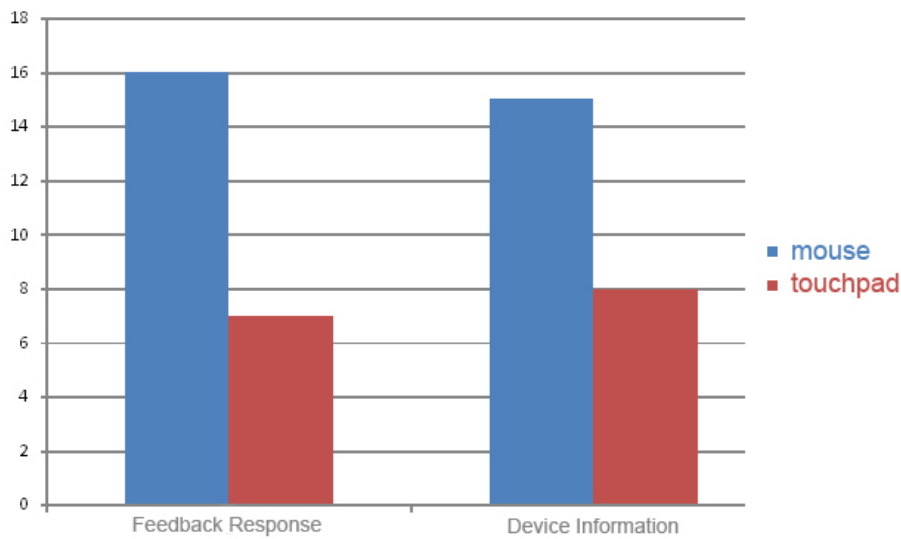


Figure 5.9: Mouse vs Touchpad pointing devices statistic from participants feedback response(left) and form collected devices information(right).

The application also tried to collect display devices information, for that it collects physical screen size, physical screen resolution, logical screen resolution, virtual screen resolution and pixel DPI of (x,y) from the participant's display devices. From the collected information we can state that minimum display device screen resolution is 1280×720 and maximum resolution is 3840×1200 . There were 6 participants out of 23 have used extended multiple displays because their logical resolution was not same as virtual resolution.

The figure 5.9 shows separated input pointing devices statistic for feedback responses and collected device information. The retrieved pointing devices information were almost same as participants feedback response.

5.5 Procedure

For the user study purpose, the application download link was provided on a website¹ and invited people to download and made trials on their machine. The application download link was posted on the social network platform and requested people to participant in the user study. The details instruction of the user study's procedure was written on the website page, starting from how to download the application to submit the feedback. There was a short video tutorial² and it was linked to the download web page to help participants for completing the user study task.

The task for the participants was simple, there was no separate work for the user study. The participants were instructed to do their regular work with their computer

¹<https://smhasanulbanna.wordpress.com/user-defined-transfer-functions-tool/>

²<https://goo.gl/B50vJN>

but before that, they just start and run our application tool. The participant can select any transfer function from the selection box or define their own transfer function by the help of interactive graph. Once the application runs, a pop-up notification message appears every 5 minutes and ask the participant to provide feedback on the currently activated transfer function. The important part of the task was to change and try few new transfer functions to experience better pointing performance. For this reason, the participants were instructed to try few different transfer functions by defining their own transfer functions and after that requested them to provide feedback based on their experience. The participants were requested to perform the same task multiple times with different customized transfer functions.

We always administered the participants during the study period for encouraging them to try new transfer function and also to remind them for making more trials and feedbacks. The participants were always connected during the whole user study period and assisted over the phone or text message in case of any help.

5.6 Result

In total there were 140 (M: 6.087, SD: 5.49) trails from 23 participants during the 3 weeks of study period. There were 114 (M: 4.96, SD: 4.86) trials out of 140 trails were the user-defined customized transfer functions, that means participants were tried to define their own transfer function and the other 26 (M: 1.13, SD: 2.47) trials were existing operating systems transfer functions (see figure 5.10).

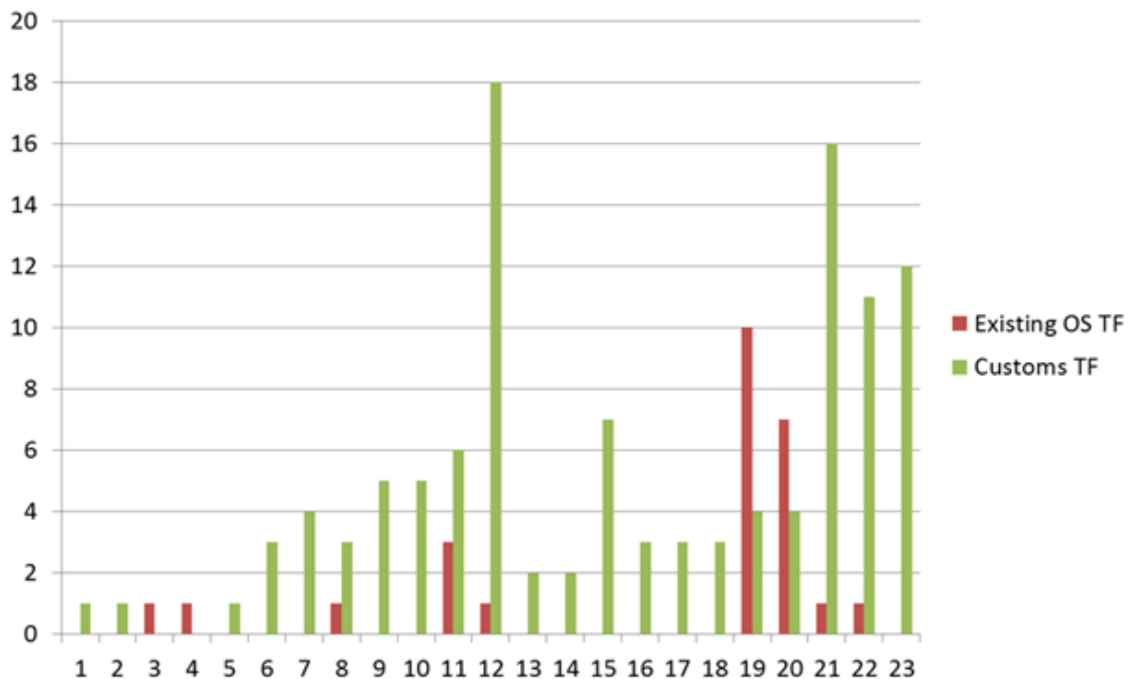


Figure 5.10: Total collected trials group by user-defined customized transfer functions and existing operating system's transfer functions.

5. Study for Interesting Transfer Function

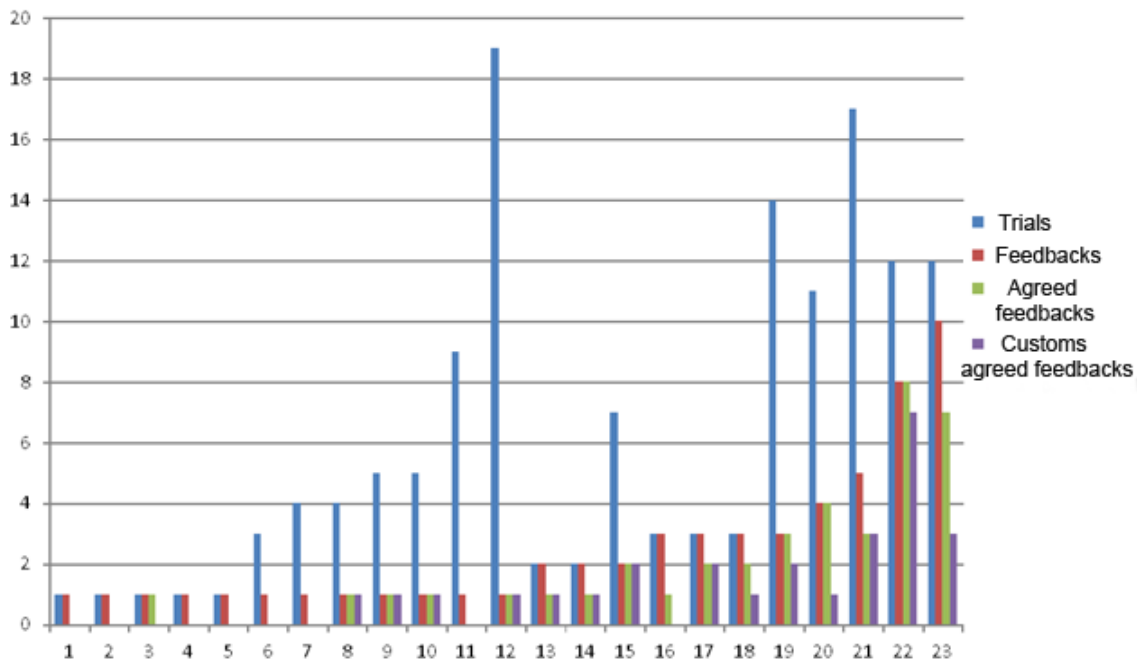


Figure 5.11: Total number of collected trials, feedbacks, marked agreed feedbacks and customized feedbacks.

There were 57 (M: 2.48, SD: 2.37) feedbacks response out of 140 trails shown in figure 5.11. Where 39 out of 57 (M: 1.69, SD: 2.19) feedbacks were evaluated as the level of agreement by the participants and they have selected positive response upon the question statement ‘*The selected transfer function fits my needs.*’. Therefore, we consider that the following transfer functions fulfill participants pointing performance demand. There were 22 out of 39 (M: 0.96, SD: 2.01) feedbacks were evaluated ‘*Strongly agree*’ and 17 out of 39 (M: 0.74, SD: 0.51) feedbacks were evaluated ‘*Agree*’ against that feedback question statement. There were 27 out of 39 (M: 1.174, SD: 1.58) positively evaluated feedbacks were user-defined customized transfer function and rest of the 12 (M: 0.52, SD: 1.04) were existing operating system’s transfer functions.

Figure 5.12 shows the statistic of the total trails and feedbacks group by pointing devices, there were 116 (M: 7.25, SD: 6.13) trails from 16 participants, who used a mouse as their most preferable pointing device and others 24 (M: 3.43, SD: 2.14) trails from 7 participants who used a touchpad as their main pointing device. There were 48 (M: 3, SD: 2.68) feedbacks provided by the mouse pointing device and only 9 (M: 1.28, SD: 0.52) feedbacks were from the touchpad.

The total time for the 140 trails was 9579.28 seconds (see figure A.9) and the average activated duration for each trails was 68.42 (SD: 167.44) seconds. The total time for the 57 feedbacks was 7580.13 seconds (see figure A.10) and average activated duration for each transfer function during feedback was 132.98 (SD: 224.24) seconds.

To identify the interesting transfer function, we have studied each of the trial and feedback provided by the participants. The evaluation parameters totally depend

on the participant's feedback response and transfer function activation time. The agreed feedback responses were evaluated and those transfer functions were considered as an interesting transfer function for the individual participant. The longest activated trial transfer function also considered as an interesting transfer function for the individual participant as participants used that transfer function for the longest period of time. Figure 5.13 shows all the transfer functions of the top 3 participants, the agreed function is plotted in green color and the longest activated functions is plotted in red color.

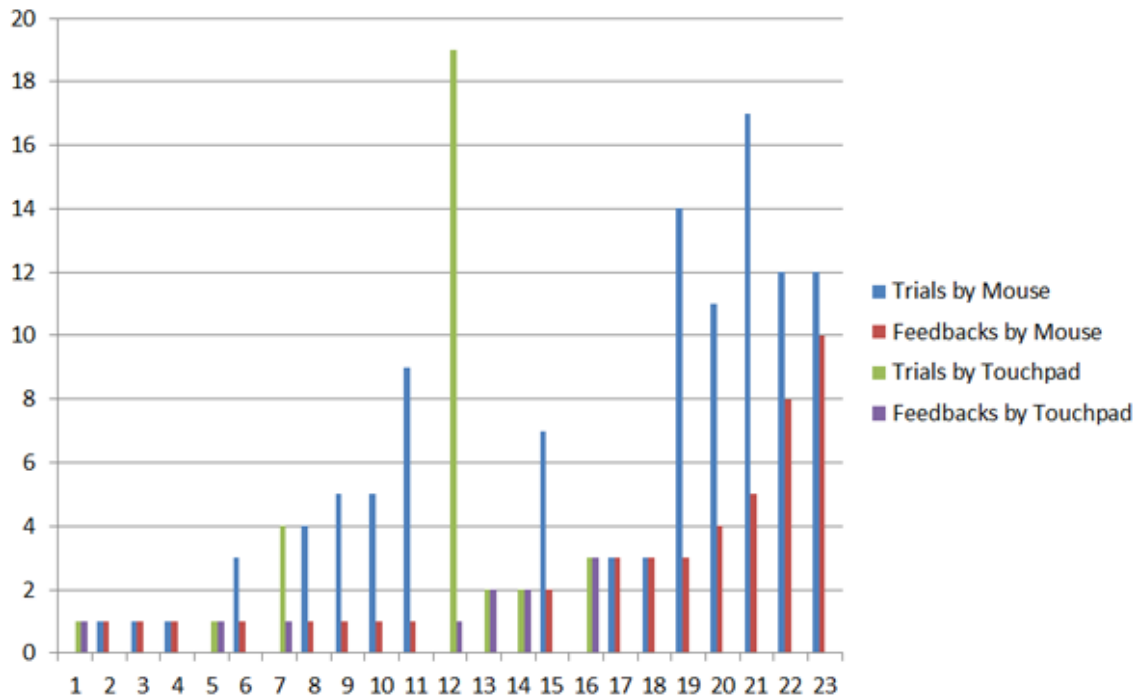
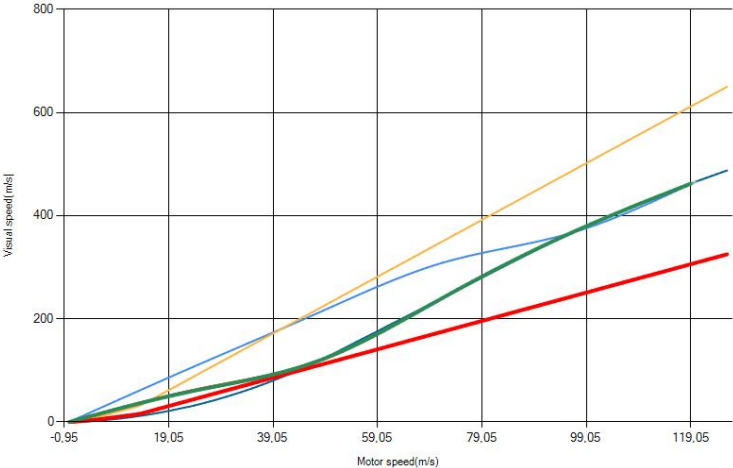


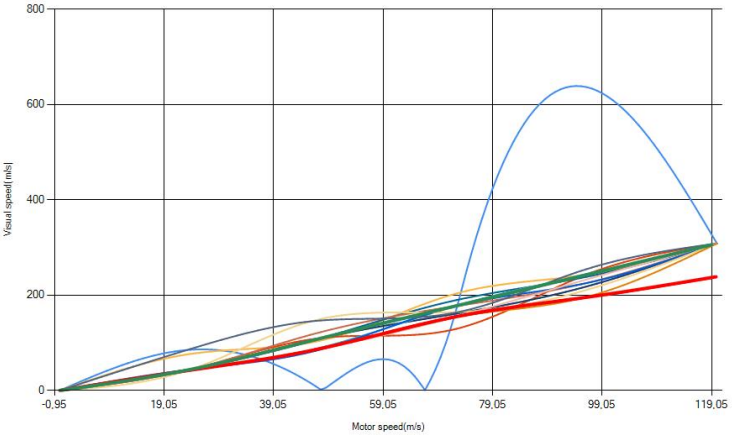
Figure 5.12: Total collected trials and feedbacks group by pointing devices.

Appendix A.6 presents a list of plotted graph of all the transfer functions tried by the participants, who has provided at least five trials and one feedback. Each figure (for example figure A.11) shows all the tried transfer functions of an individual participant and the agreed transfer function plotted in green color and the longest activated transfer functions plotted in red color.

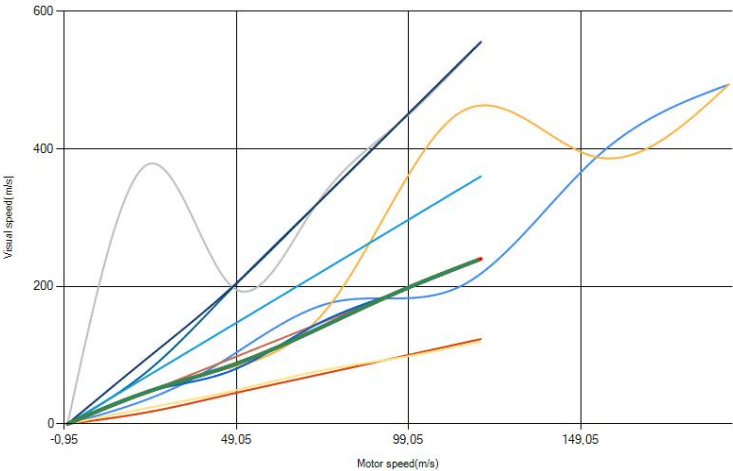
5. Study for Interesting Transfer Function



(a) Participant No. 19



(b) Participant No. 21



(c) Participant No. 12

Figure 5.13: Interesting Transfer Function for individual participants (agreed transfer function plotted in green and longest activated function plotted in red).

5.7 Discussion

Now to find out the interesting transfer function from the overall user study result, we have decided to take the agreed response transfer functions and the longest trial transfer functions as our evaluated parameters. Therefore, the identified interesting transfer function will be based on feedback response and trial activation duration. The collected trails and feedbacks are from two different pointing device such as mouse and touchpad. So Finally we have decided to identify four interesting transfer function, such as

1. Interesting transfer function for mouse based on participant's feedback:

To identify this transfer function we have considered all the '*Agreed*' and '*Strongly agreed*' evaluated feedbacks provided by the mouse pointing device. After that, we have selected those transfer function's curve and took six pairs of data points from each transfer function curve according to the knot position of our interactive graph system. We have generated a box diagram(see figure 5.14a) from those collected six-knot data points sets. After selecting all the median values from the generated box diagram, we have drawn the interesting transfer function(see figure 5.14b) through the median values by the help of Spline Interpolation function. Figure 5.14 shows the graphs of the interesting transfer function for the mouse based on participant's feedback response. More details and statistic of the box plot diagram is given in the appendix A.21.

2. Interesting transfer function for mouse based on participant's trial duration:

To identify this interesting transfer function we have selected the longest trial for each participant. There are 15 out of 23 participants used a mouse as a pointing device, so we took 15 transfer function curves. After that, we have selected six pairs of data points from that 15 transfer function curves, according to the knot position of our interactive graph system. Then we have generated a box diagram(see figure 5.15a) from those collected six-knot data points sets. After that, we have selected all the median values from the generated box diagram and plotted the interesting transfer function(see figure 5.15b) through the medians by the help of Spline Interpolation function. Figure 5.15 shows the graphs of the interesting transfer function for the mouse based on trial activation time. More details and statistic of the box plot diagram is given in the appendix A.23.

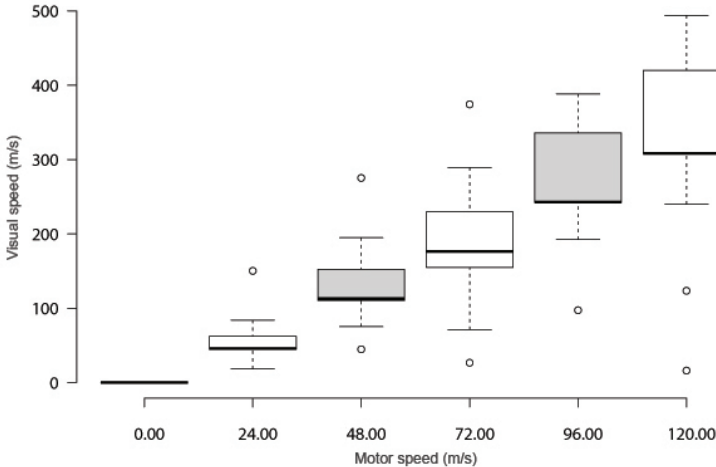
3. Interesting transfer function for touchpad based on participant's feedback:

To identify this interesting transfer function we followed that same procedure of the Mouse interesting transfer function based on participant's feedback response but this time we considered those response provided by the touchpad. Figure 5.16a shows the box plot diagram for this case and figure 5.16b shows the interesting transfer function drawn through the median values of the box diagram. Figure 5.16 shows

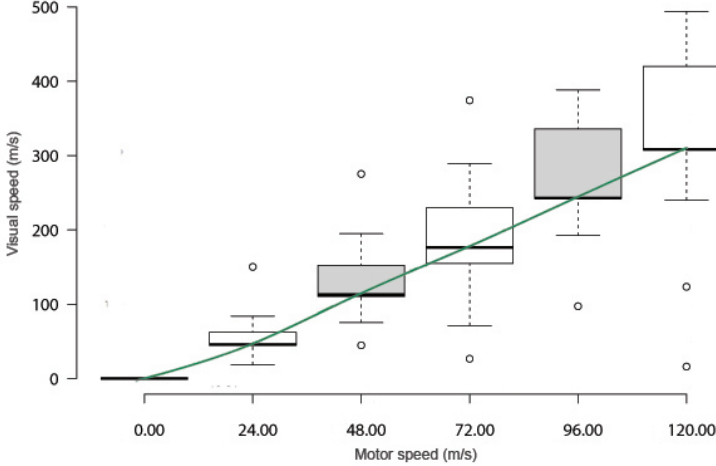
the interesting transfer function for the touchpad based on participant's feedback response. More details and statistic of the box plot diagram is given in the appendix A.22.

4. Interesting transfer function for touchpad based on participant's trial duration:

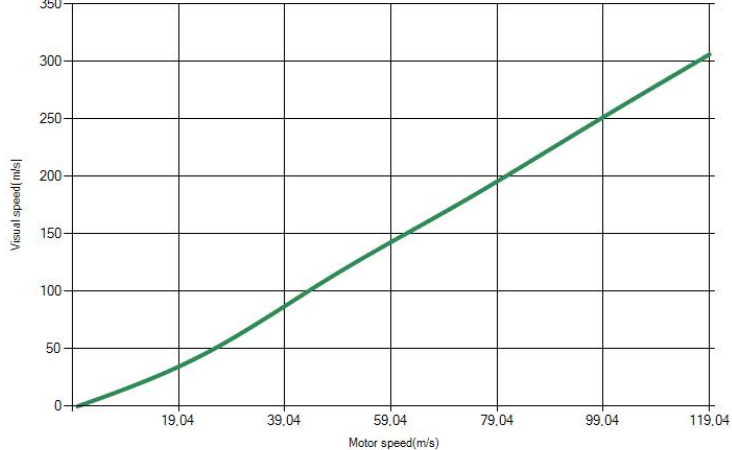
To identify this interesting transfer function we followed the same procedure which we used for the Mouse interesting transfer function based on participants trial duration but the only difference is that this time we have considered those trials provided by the touchpad. Figure 5.17a shows the box plot diagram for this case and figure 5.17b shows the interesting transfer function drawn through the median values of the box diagram. Figure 5.17 shows the interesting transfer function for the touchpad based on trial activation time. More details and statistic of the box plot diagram is given in the appendix A.24.



(a) Box Plot Diagram



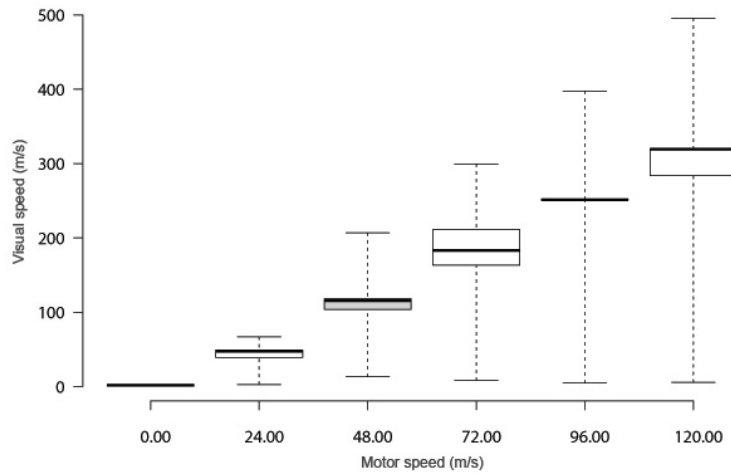
(b) Interesting transfer function draw through the median values



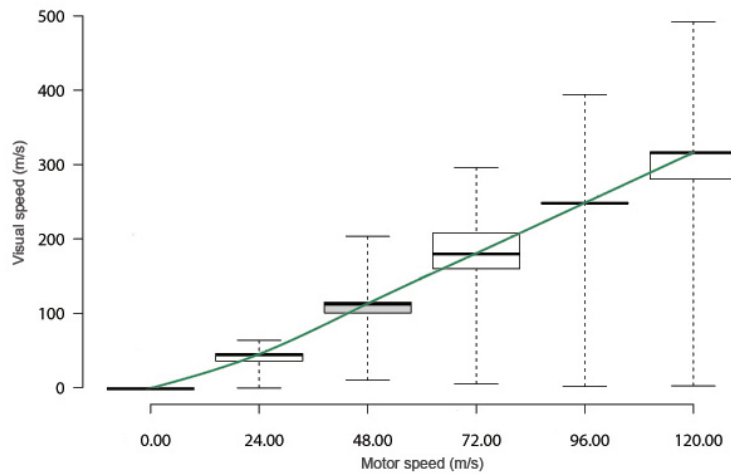
(c) Interesting Transfer Function

Figure 5.14: The interesting transfer function for the mouse based on participant's feedback response.

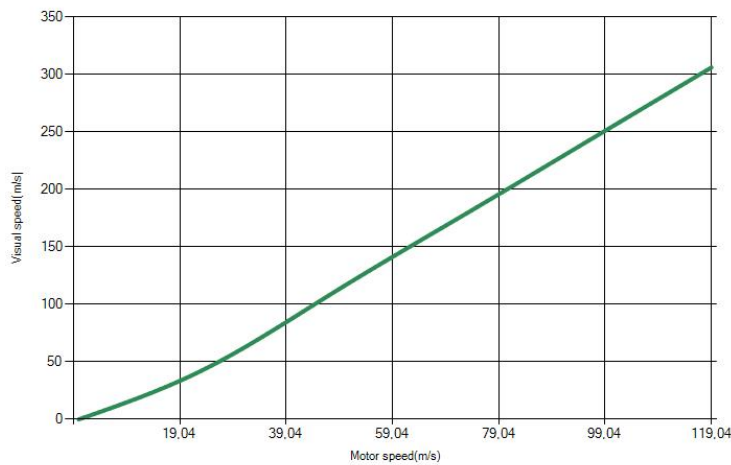
5. Study for Interesting Transfer Function



(a) Box Plot Diagram

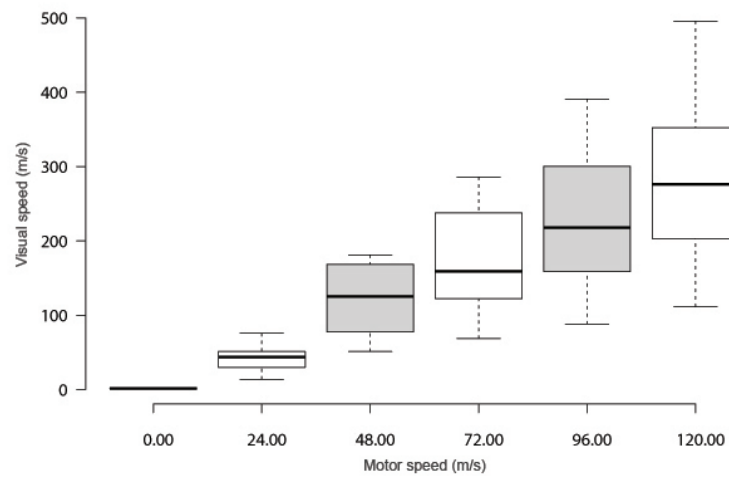


(b) Interesting transfer function draw through the median values

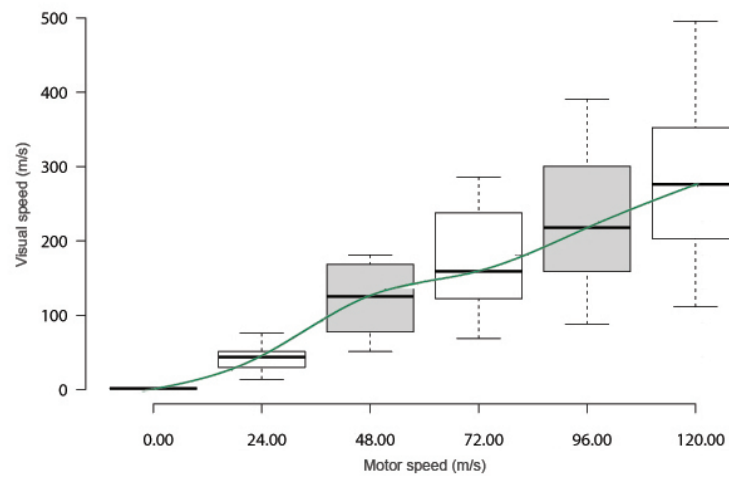


(c) Interesting Transfer Function

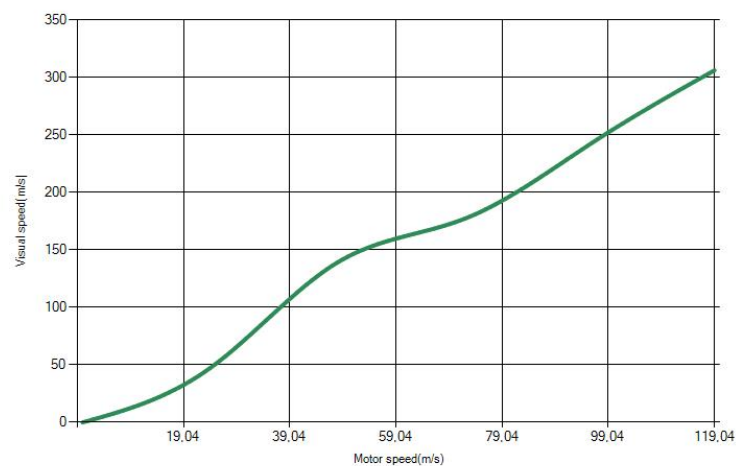
Figure 5.15: The interesting transfer function for the mouse based on participant's trial activation time.



(a) Box Plot Diagram



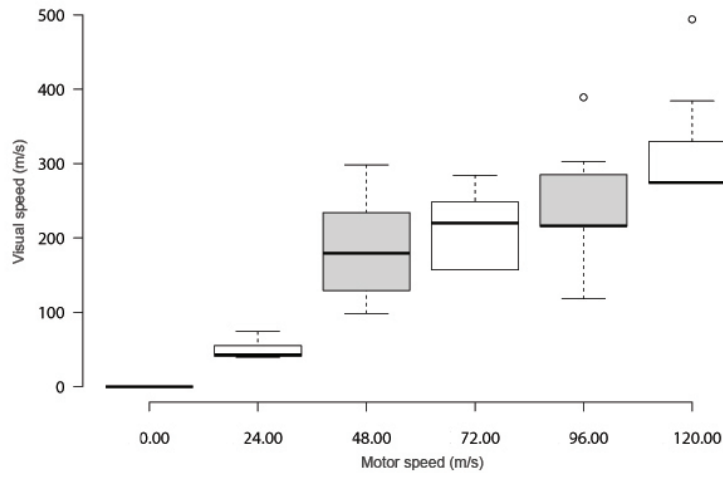
(b) Interesting transfer function draw through the median values



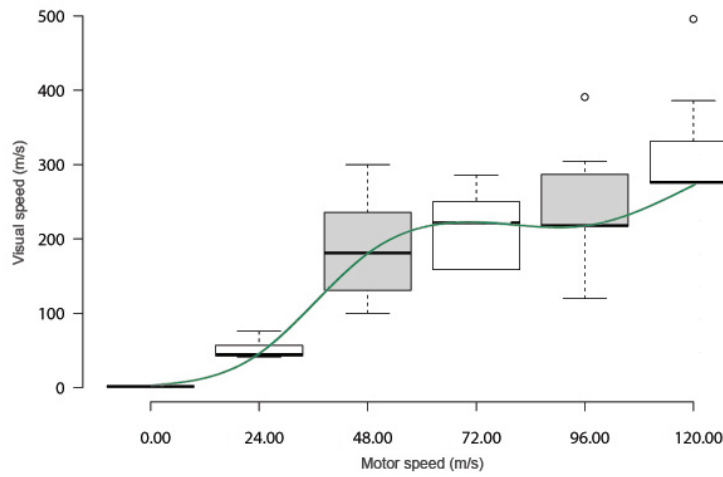
(c) Interesting Transfer Function

Figure 5.16: The interesting transfer function for the touchpad based on participant's feedback response.

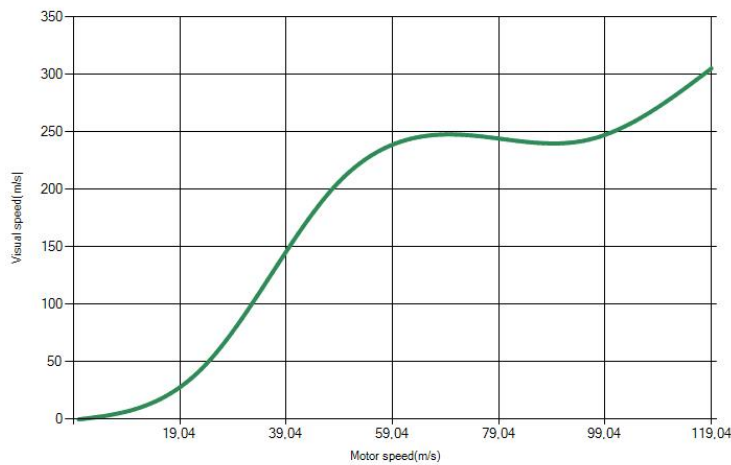
5. Study for Interesting Transfer Function



(a) Box Plot Diagram



(b) Interesting transfer function draw through the median values



(c) Interesting Transfer Function

Figure 5.17: The interesting transfer function for the touchpad based on participant's trial activation time.

5.8 Summary

In total there were 140 trails and 57 feedbacks from 23 participants within 3 weeks of user study period, which was not a poor response at all but our acceptance was little more. The participant's involvements with the application and the total number of user-defined transfer functions were quite impressive. In total 114 new transfer functions were collected from the user study. The initial plan was to identify interesting transfer function for each participant. Later the plan was changed and the interesting transfer functions are identified from the overall user study statistic for mouse and touchpad. As we know that these two pointing devices are the best match for human needs and pointing performances for the graphical interface. The identification of the interesting transfer functions is totally depended on the participant's activities and feedback response.

6 | Conclusion and Future Work

There are million of people all over the world using pointing device without any sort of technical knowledge how the inside mechanism work. They have no idea about the pointing performance and evaluating parameters. People just relied on the arbitrary satisfaction after changing the pointer setting of their computer. The public documentations of the pointing technique from most of the modern operating system is obscure. Therefore, this thesis presents a detailed description of the most important pointing transfer functions to gain a deeper understanding of the optimal transfer function. We have also described some popular pointing devices. The idea of the thesis begins with the implementation of collecting raw input data of the pointing devices. After that, we have tried to make a mouse recorder which will record all the raw information from any pointing devices. The recorder also records all the necessary and sufficient information related to the pointer on the display to understand a pointing transfer function.

During the progress of the thesis, the main idea was buildup. The main objective of the thesis is to assess the pointing transfer function based on natural interaction with the system using a user-driven approach. To implement this part we have developed an application tool where a user can select a transfer function from the selection box and define their own transfer function. To replicate all the existing modern transfer functions we have used data points sets from the *libpointing* toolkit repository and that makes our implementation easier.

The challenging part was to find out an innovative and convenient way to manipulate a range of data points for defining user own transfer function, which also needs to be relevant in sense of pointing transfer function curve. For this reason, we have introduces an interactive graph system, where a user can define their own transfer function by just a mouse drag. The complete desktop application tool is the contribution of our thesis.

After finishing the implementation part, a user study was conducted to identify interesting transfer function. The interesting transfer functions are identified based on participants feedback response and trials duration for the mouse and touchpad pointing devices. Finally, we have identified four interesting transfer function, two of them are based on participants feedback response and others two of them are based on trial activation time. The identified transfer functions contain basic characteristic such as when pointing device velocity is high the cursor moves faster and when the

pointing device velocity is low the cursor moves slower.

Future work, the idea could be deployed in different perspectives and the desktop tool could be extended by integrating additional features. One idea could be to observe the user's pointing devices input behavior and cursor movements behavior over a period of time and then propose an optimal transfer function without requiring feedback from the users. The comparison between different transfer functions could be another option. There are others options such as evaluating or measuring the performance of the optimal transfer function based on accuracy, precision, target hit prediction time and difficulty level and so on. Most of the time users have no idea about the pointing performance measurement and the evaluating parameter such as accuracy, precision, target hit time and difficulty level. They relied only on their tentative experience. Therefore, if the user could see the evaluations in parameters of the selected transfer function after changing any pointer setting then it will help them to improve their pointing performance in graphical user interfaces.

Bibliography

- [1] Pointer ballistics for windows xp. *Archived white paper, Windows Hardware Developer Center*, Oct. 2002.
- [2] Renaud Blanch, Yves Guiard, and Michel Beaudouin-Lafon. Semantic pointing: improving target acquisition with control-display ratio adaptation. pages 519–526, 2004.
- [3] Universal Serial Bus. Device class definition for human interface devices (hid). *Version, 1:1996–2001*, 2001.
- [4] William Buxton, Mark Billingham, Yves Guiard, Abigail Sellen, and Shumin Zhai. Human input to computer systems: theories, techniques and technology. *Manuscrito de livro em andamento, sem editora*, 2002.
- [5] Stuart K Card, William K English, and Betty J Burr. Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a crt. *Ergonomics*, 21(8):601–613, 1978.
- [6] Géry Casiez and Nicolas Roussel. No more bricolage!: methods and tools to characterize, replicate and compare pointing transfer functions. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 603–614. ACM, 2011.
- [7] Géry Casiez, Daniel Vogel, Ravin Balakrishnan, and Andy Cockburn. The impact of control-display gain on user performance in pointing tasks. *Human-Computer Interaction*, 23(3):215–250, 2008.
- [8] ERFW Crossman and PJ Goodeve. Feedback control of hand-movement and fitts’ law. *The Quarterly Journal of Experimental Psychology*, 35(2):251–278, 1983.
- [9] Sarah A. Douglas and Anant Kartik Mithal. *The Ergonomics of Computer Pointing Devices*. Springer-Verlag London Limited, 1997.
- [10] D.C. Engelbart, W. K. English, and B. Huddart. *Computer-aided display control Final report*. NASA Technical Documents, 1965.

- [11] Paul M Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of experimental psychology*, 47(6):381, 1954.
- [12] CB Gibbs. Controller design: Interactions of controlling limbs, time-lags and gains in positional and velocity systems. *Ergonomics*, 5(2):385–402, 1962.
- [13] Ken Hinckley and Daniel Wigdor. Input technologies and techniques. *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications*, pages 151–168, 2002.
- [14] Denis Howe. Free on-line dictionary of computer, 1985. URL <http://foldoc.org/trackpad>.
- [15] Herbert D Jellinek and Stuart K Card. Powermice and user performance. pages 213–220, 1990.
- [16] Eike Jessen, Dieter Michel, and Heinz Voigt. Structure, technology, and development of the aeg-telefunken tr 440 computer. *IEEE Annals of the History of Computing*, 32(3):30–39, 2010.
- [17] Steven W Keele. Movement control in skilled motor performance. *Psychological bulletin*, 70(6p1):387, 1968.
- [18] Steve Kolokowsky and T Davis. Touchscreens 101: Understanding touchscreen technology and design. *Cypress Semiconductor Corp. California*, 2009.
- [19] Richard F. Lyon. *The Optical Mouse, and an Architectural Methodology for Smart Digital Sensors*. Xerox, Palo Alto Research Center, 1981.
- [20] I Scott MacKenzie. Fitts’ law as a research and design tool in human-computer interaction. *Human-computer interaction*, 7(1):91–139, 1992.
- [21] I Scott MacKenzie and Poika Isokoski. Fitts’ throughput and the speed-accuracy tradeoff. pages 1633–1636, 2008.
- [22] Larry Masinter, Tim Berners-Lee, and Roy T Fielding. Uniform resource identifier (uri): Generic syntax. 2005.
- [23] Michael McGuffin and Ravin Balakrishnan. Acquisition of expanding targets. pages 57–64, 2002.
- [24] David E Meyer, Richard A Abrams, Sylvan Kornblum, Charles E Wright, and JE Keith Smith. Optimality in human motor performance: Ideal control of rapid aimed movements. *Psychological review*, 95(3):340, 1988.
- [25] Bill Moggridge and Bill Atkinson. Designing interactions. volume 14. MIT press Cambridge, MA, 2007.
- [26] David A Rosenbaum. Human motor control. 1991.

- [27] Mark S Sanders and Ernest J McCormick. Human factors in engineering and design. 1987.
- [28] Richard A Schmidt, Howard Zelaznik, Brian Hawkins, James S Frank, and John T Quinn Jr. Motor-output variability: A theory for the accuracy of rapid motor acts. *Psychological review*, 86(5):415, 1979.
- [29] Miika Silfverberg, I Scott MacKenzie, and Tatu Kauppinen. An isometric joystick as a pointing device for handheld information terminals. 2001:119–126, 2001.
- [30] Jacob O Wobbrock, James Fogarty, Shih-Yen Sean Liu, Shunichi Kimuro, and Susumu Harada. The angle mouse: target-agnostic dynamic gain adjustment based on angular deviation. pages 1401–1410, 2009.

List of Figures

2.1	Generic block diagram of a pointing device (from [9]).	14
2.2	Possible sequence(s) of submovements toward a target as described by the optimized initial impulse model [26]. (a) is the case where a single movement reaches the target. (b) and (c) are the more likely cases where the initial movement under or over shoots the target, requiring subsequent corrective movements (from [23]).	20
2.3	Parent transfer function graph of Windows XP with five points, and zoom view of four points are on the right side(from [1]).	21
2.4	Windows 10 configuration interface with default settings. Same setting used by other old version of windows.	22
2.5	Registry Editor- Mouse Lookup Table, Microsoft Windows 10; Version 1607, 2016.	23
2.6	Ubuntu 12.04 mouse pointer configuration interface, same setting used by others version.	25
2.7	Xorg functions available in Ubuntu 10.10 through the interface shown in Figure 2.6 (from [6]).	26
2.8	OS X 10.6.7 configuration interface for the mouse. A tooltip associated to the slider says <i>Drag to adjust how fast you want the pointer to follow the movement of your mouse.</i>	27
2.9	OS X 10.6.7 functions for mice (available through the interface shown in Figure 2.8) and touchpads (from [6]).	27
3.1	Application Process Diagram.	31
4.1	The simple architecture of the MouseLogger Tool.	38
4.2	The screenshot of the Mouse Recorder Application Tool.	39
4.3	The pop-up notification message (Left) and System Tray Icon right click menu options (Right).	40
4.4	The simple Architecture of the User-defined Transfer Functions Tool.	41
4.5	The screenshot of the User-defined Transfer Function tool main interface.	42
4.6	The screenshot of the selection combo box.	43
4.7	The Application main menu (Left: File Save and Open Option, Right: Feedback Option).	43

4.8	Start notification message (Left) and System Tray Icon right click menu options (Right).	44
4.9	File structure system of the libpointing data points.	45
5.1	Feedback options in the main menu.	49
5.2	Demographic information dialog box.	50
5.3	Pop-up notification message box reminds the participants for providing feedback about the currently activated transfer function.	51
5.4	Feedback questionnaire dialog box for user study purpose.	52
5.5	User Information text file.	53
5.6	Feedback log file.	53
5.7	The application architecture after adding Google spreadsheet.	54
5.8	GoogleSheets tables and Entity-Relationship (ER) model.	55
5.9	Mouse vs Touchpad pointing devices statistic from participants feedback response(left) and form collected devices information(right).	56
5.10	Total collected trials group by user-defined customized transfer functions and existing operating system's transfer functions.	57
5.11	Total number of collected trials, feedbacks, marked agreed feedbacks and customized feedbacks.	58
5.12	Total collected trials and feedbacks group by pointing devices.	59
5.13	Interesting Transfer Function for individual participants (agreed transfer function plotted in green and longest activated function plotted in red).	60
5.14	The interesting transfer function for the mouse based on participant's feedback response.	63
5.15	The interesting transfer function for the mouse based on participant's trial activation time.	64
5.16	The interesting transfer function for the touchpad based on participant's feedback response.	65
5.17	The interesting transfer function for the touchpad based on participant's trial activation time.	66
A.1	Mouse Logger log file screen shot (partial).	84
A.2	Screen shot of user-defined application tool's log file(partial).	85
A.3	Screen shot of interactive user-defined transfer function graph's log file(partial).	86
A.4	Base class diagram of Mouse Logger application tool.	87
A.5	Base class diagram of User-defined transfer function application tool.	88
A.6	Screen short of Google sheets - customized transfer functions.	90
A.7	Screen short of Google sheets - User defined transfer function feedback[part 1].	91
A.8	Screen short of Google sheets - User defined transfer function feedback[part 2].	92
A.9	Trails transfer function activated time.	95
A.10	Feedback transfer function activated time.	95

A.11 Interesting Transfer Function for Participants User9(best fits marked green and longest activated trial marked red).	97
A.12 Interesting Transfer Function for Participants User10(best fits marked green and longest activated trial marked red).	97
A.13 Interesting Transfer Function for Participants User11(best fits marked green and longest activated trial marked red).	98
A.14 Interesting Transfer Function for Participants User12(best fits marked green and longest activated trial marked red).	98
A.15 Interesting Transfer Function for Participants User15(best fits marked green and longest activated trial marked red).	99
A.16 Interesting Transfer Function for Participants User19(best fits marked green and longest activated trial marked red).	99
A.17 Interesting Transfer Function for Participants User20(best fits marked green and longest activated trial marked red).	100
A.18 Interesting Transfer Function for Participants User21(best fits marked green and longest activated trial marked red).	100
A.19 Interesting Transfer Function for Participants User22(best fits marked green and longest activated trial marked red).	101
A.20 Interesting Transfer Function for Participants User23(best fits marked green and longest activated trial marked red).	101
A.21 Box plot graph for mouse interesting transfer function based on agreed feedback response.	102
A.22 Box plot graph for touchpad interesting transfer function based on agreed feedback response.	103
A.23 Box plot graph for mouse interesting transfer function based on trial activated time.	104
A.24 Box plot diagram statistics for the mouse interesting transfer function based on trial activated time.	105
A.25 Interesting transfer function for mouse based on agreed feedback response.	106
A.26 Interesting transfer function for touchpad based on agreed feedback response.	106
A.27 Interesting transfer function for mouse based on trial activated time. .	107
A.28 Interesting transfer function for touchpad based on trial activated time.	107

List of Tables

2.1	Windows mouse sensitivity when position multiplier enhance pointer precision is off & on	24
A.1	Windows Transfer Function data points set when EPP on and Speed 6.	94
A.2	Box plot diagram statistics for the mouse interesting transfer function based on agreed feedback response.	102
A.3	Box plot diagram statistics for the touchpad interesting transfer function based on agreed feedback response.	103
A.4	Box plot statistics for mouse interesting transfer function based on trial activated time.	104
A.5	Box plot diagram statistics for the touchpad interesting transfer function based on trial activated time.	105

Abbreviations

GUIs	Graphical User Interfaces
CD	Control Display
PA	Pointer Acceleration
OS	Operating System
USB	Universal Serial Bus
HID	Human Interface Devices
A-D	Analog-to-digital
UIMS	User Interface Management System
LED	Light Emitting Diode
LCD	Liquid Crystal Display
DPI	Dot Per Inch
PIC	Peripheral Interface Controller
URI	Uniform Resource Identifier
API	Application programming interface
CPI	Count per Inch
CSV	Comma separated value
GUI	Graphical User Interface
EPP	Enhanced pointer precision
UI	User Interface
ER	Entity-Relationship

A | Appendix

A.1 Log File Screen Shot

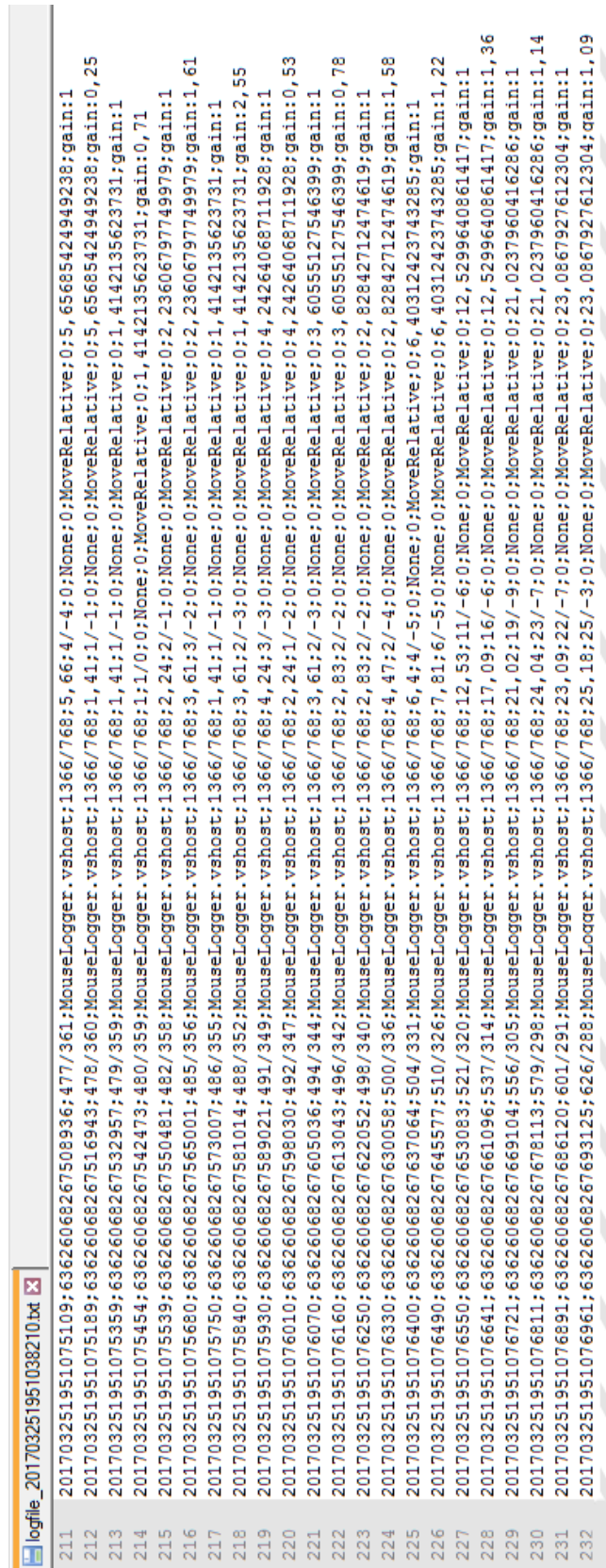
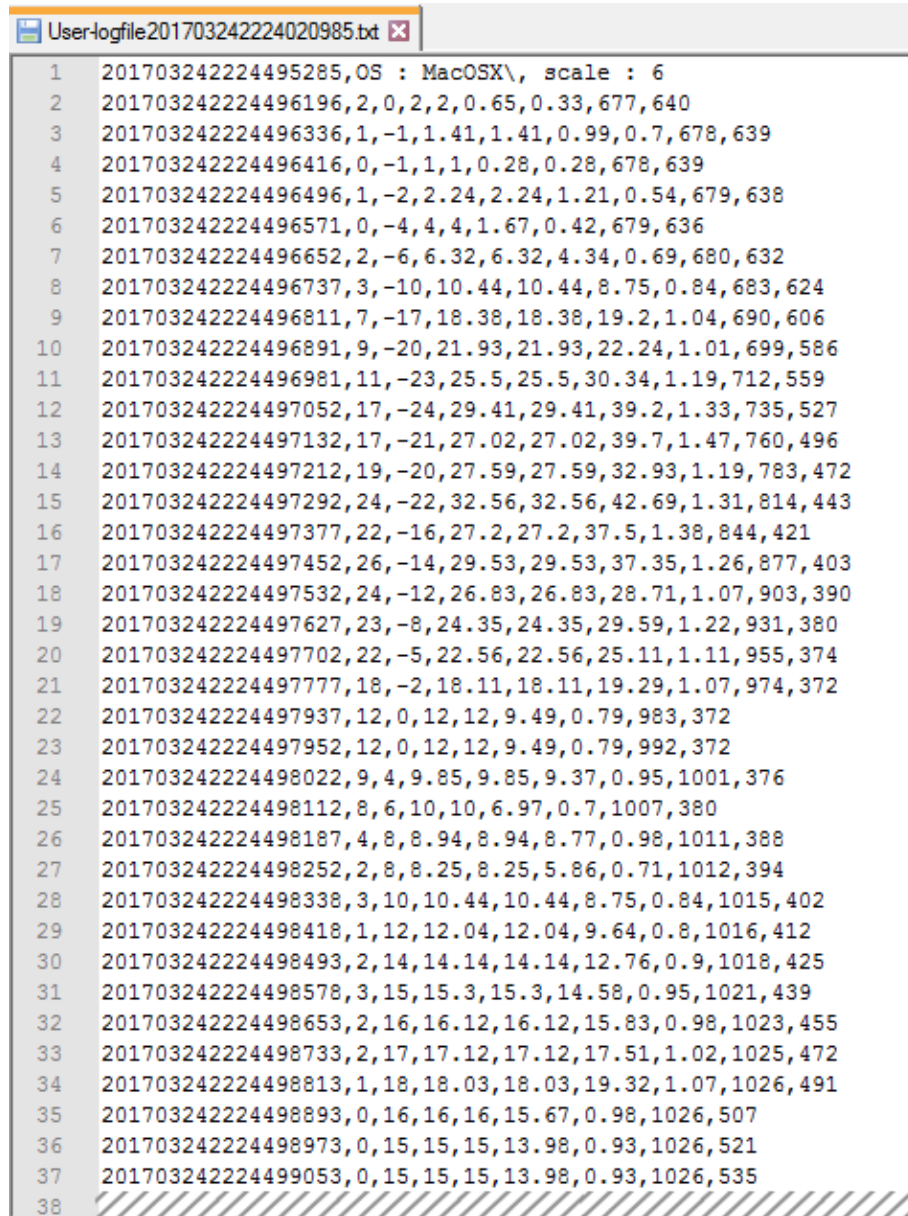
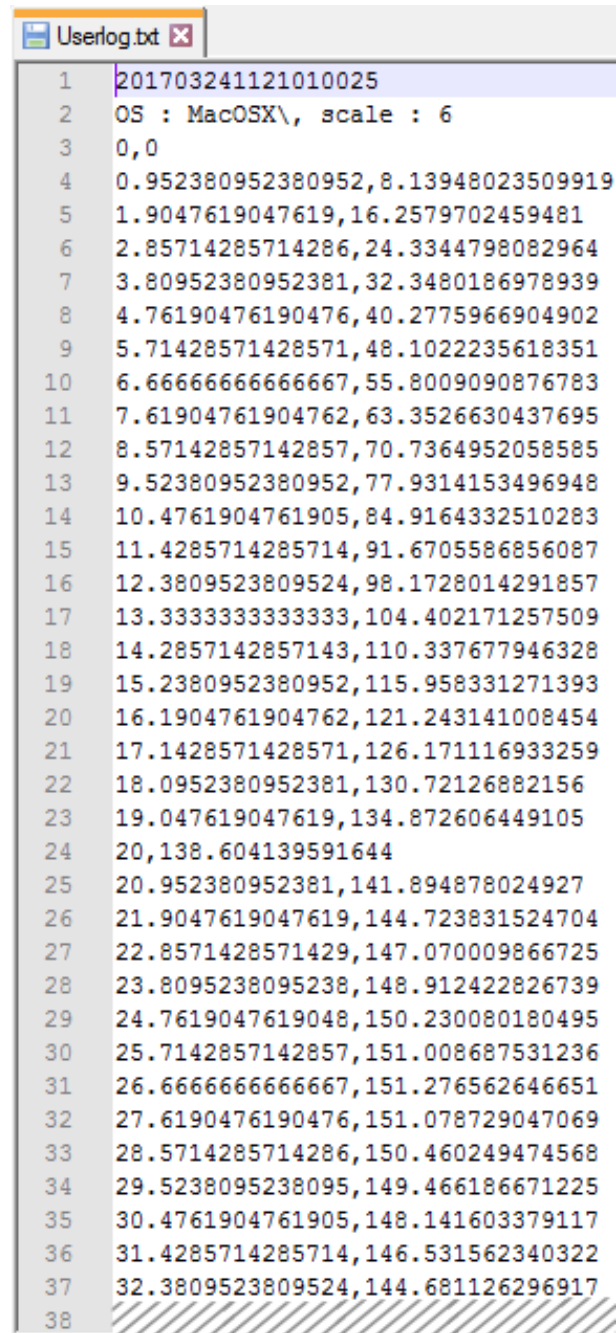


Figure A.1: Mouse Logger log file screen shot (partial).



```
User-logfile201703242224020985.txt
1 201703242224495285,OS : MacOSX\, scale : 6
2 201703242224496196,2,0,2,2,0.65,0.33,677,640
3 201703242224496336,1,-1,1.41,1.41,0.99,0.7,678,639
4 201703242224496416,0,-1,1,1,0.28,0.28,678,639
5 201703242224496496,1,-2,2.24,2.24,1.21,0.54,679,638
6 201703242224496571,0,-4,4,4,1.67,0.42,679,636
7 201703242224496652,2,-6,6.32,6.32,4.34,0.69,680,632
8 201703242224496737,3,-10,10.44,10.44,8.75,0.84,683,624
9 201703242224496811,7,-17,18.38,18.38,19.2,1.04,690,606
10 201703242224496891,9,-20,21.93,21.93,22.24,1.01,699,586
11 201703242224496981,11,-23,25.5,25.5,30.34,1.19,712,559
12 201703242224497052,17,-24,29.41,29.41,39.2,1.33,735,527
13 201703242224497132,17,-21,27.02,27.02,39.7,1.47,760,496
14 201703242224497212,19,-20,27.59,27.59,32.93,1.19,783,472
15 201703242224497292,24,-22,32.56,32.56,42.69,1.31,814,443
16 201703242224497377,22,-16,27.2,27.2,37.5,1.38,844,421
17 201703242224497452,26,-14,29.53,29.53,37.35,1.26,877,403
18 201703242224497532,24,-12,26.83,26.83,28.71,1.07,903,390
19 201703242224497627,23,-8,24.35,24.35,29.59,1.22,931,380
20 201703242224497702,22,-5,22.56,22.56,25.11,1.11,955,374
21 201703242224497777,18,-2,18.11,18.11,19.29,1.07,974,372
22 201703242224497937,12,0,12,12,9.49,0.79,983,372
23 201703242224497952,12,0,12,12,9.49,0.79,992,372
24 201703242224498022,9,4,9.85,9.85,9.37,0.95,1001,376
25 201703242224498112,8,6,10,10,6.97,0.7,1007,380
26 201703242224498187,4,8,8.94,8.94,8.77,0.98,1011,388
27 201703242224498252,2,8,8.25,8.25,5.86,0.71,1012,394
28 201703242224498338,3,10,10.44,10.44,8.75,0.84,1015,402
29 201703242224498418,1,12,12.04,12.04,9.64,0.8,1016,412
30 201703242224498493,2,14,14.14,14.14,12.76,0.9,1018,425
31 201703242224498578,3,15,15.3,15.3,14.58,0.95,1021,439
32 201703242224498653,2,16,16.12,16.12,15.83,0.98,1023,455
33 201703242224498733,2,17,17.12,17.12,17.51,1.02,1025,472
34 201703242224498813,1,18,18.03,18.03,19.32,1.07,1026,491
35 201703242224498893,0,16,16,16,15.67,0.98,1026,507
36 201703242224498973,0,15,15,15,13.98,0.93,1026,521
37 201703242224499053,0,15,15,15,13.98,0.93,1026,535
38 /
```

Figure A.2: Screen shot of user-defined application tool's log file(partial).



```
1 201703241121010025
2 OS : MacOSX\, scale : 6
3 0,0
4 0.952380952380952,8.13948023509919
5 1.9047619047619,16.2579702459481
6 2.85714285714286,24.3344798082964
7 3.80952380952381,32.3480186978939
8 4.76190476190476,40.2775966904902
9 5.71428571428571,48.1022235618351
10 6.66666666666667,55.8009090876783
11 7.61904761904762,63.3526630437695
12 8.57142857142857,70.7364952058585
13 9.52380952380952,77.9314153496948
14 10.4761904761905,84.9164332510283
15 11.4285714285714,91.6705586856087
16 12.3809523809524,98.1728014291857
17 13.3333333333333,104.402171257509
18 14.2857142857143,110.337677946328
19 15.2380952380952,115.958331271393
20 16.1904761904762,121.243141008454
21 17.1428571428571,126.171116933259
22 18.0952380952381,130.72126882156
23 19.047619047619,134.872606449105
24 20,138.604139591644
25 20.952380952381,141.894878024927
26 21.9047619047619,144.723831524704
27 22.8571428571429,147.070009866725
28 23.8095238095238,148.912422826739
29 24.7619047619048,150.230080180495
30 25.7142857142857,151.008687531236
31 26.6666666666667,151.276562646651
32 27.6190476190476,151.078729047069
33 28.5714285714286,150.460249474568
34 29.5238095238095,149.466186671225
35 30.4761904761905,148.141603379117
36 31.4285714285714,146.531562340322
37 32.3809523809524,144.681126296917
38
```

Figure A.3: Screen shot of interactive user-defined transfer function graph's log file(partial).

A.2 Class Diagram

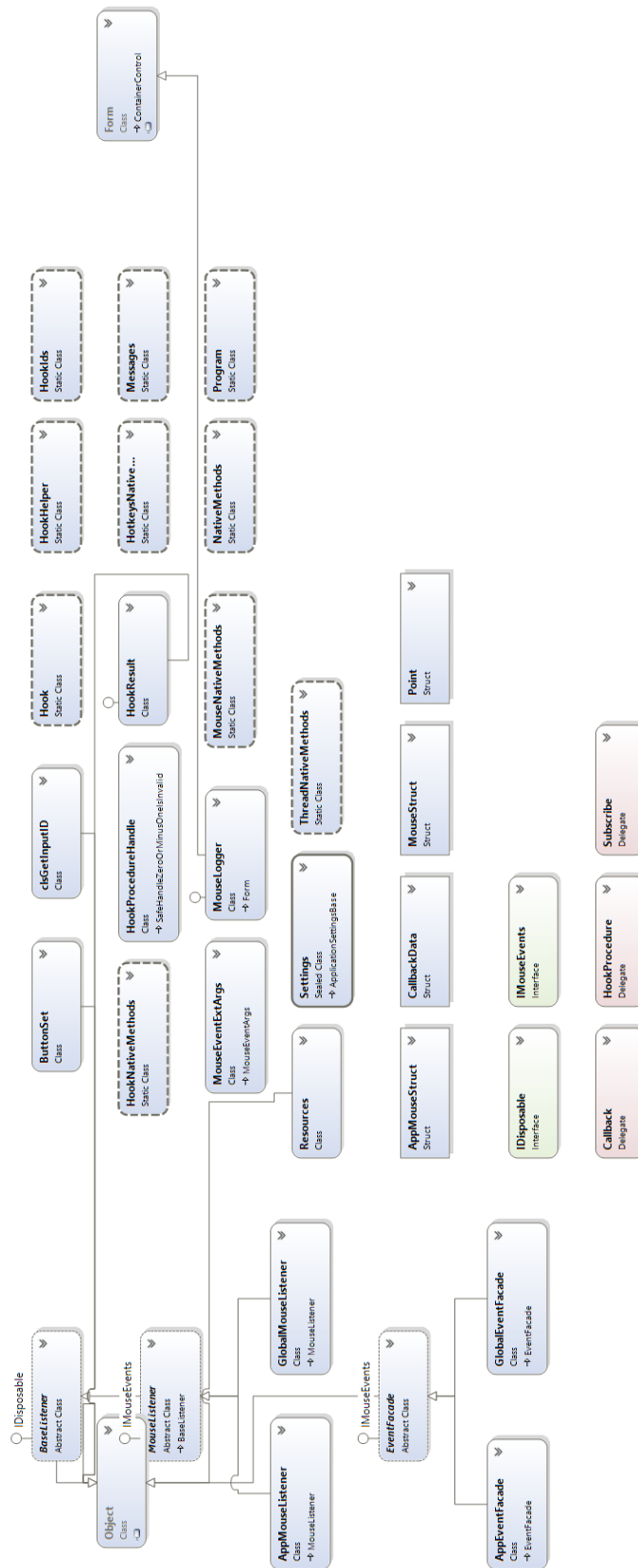


Figure A.4: Base class diagram of Mouse Logger application tool.

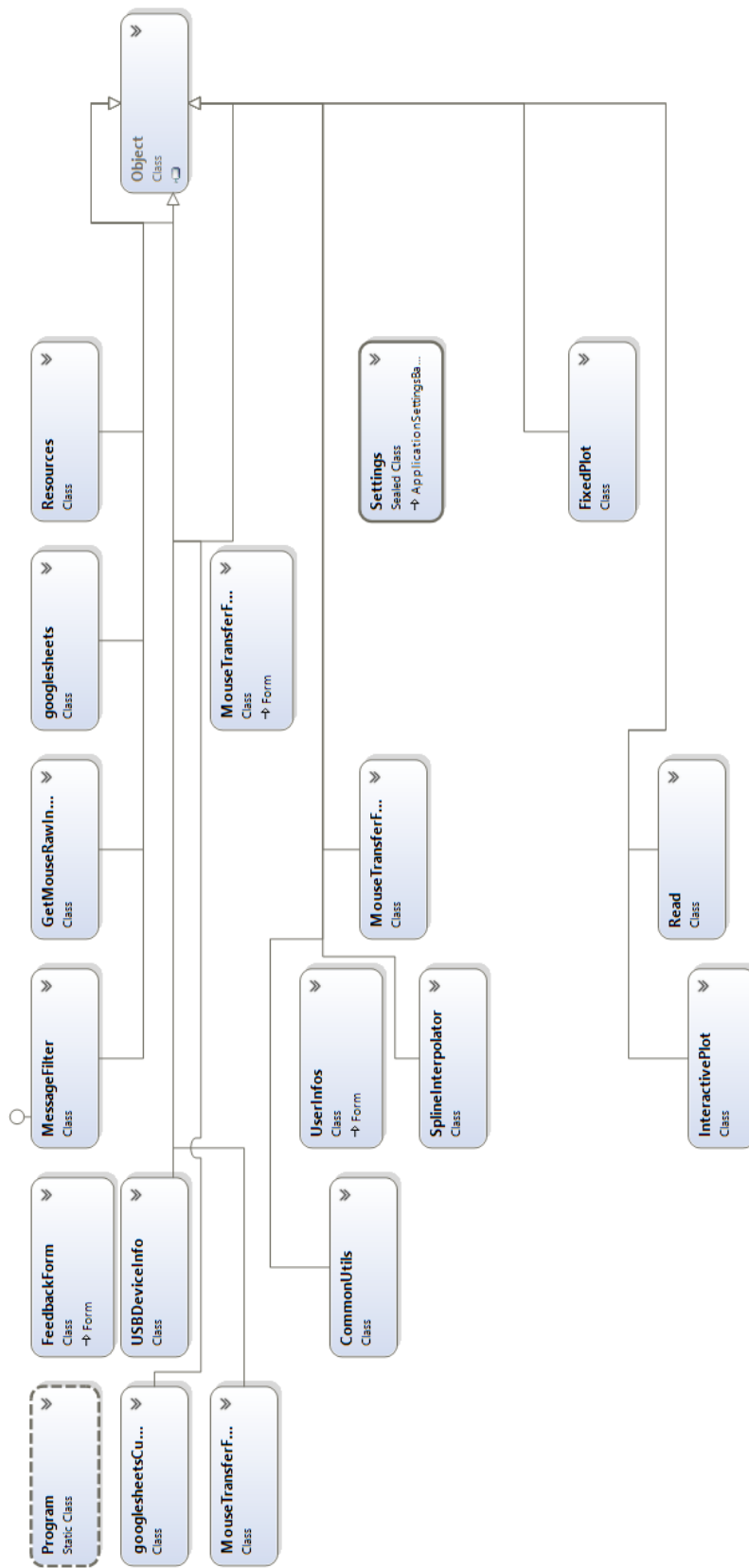


Figure A.5: Base class diagram of User-defined transfer function application tool.

A.3 Google Sheets

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	timestamp	pointingDevices	displayInfos	userID	age	sex	profession	useMouse	useTouchpad	activatedTime	selectedTFname	TFSpeed	CustomTF
2	03.23.17 22:32:16	DeviceID: USBVID_046D:0952	PhysicalScreenSize 309x174	201703232232001881						4.46s	Windows_EPP\	6	
3	03.23.17 22:33:05	DeviceID: USBVID_046D:0952	PhysicalScreenSize 309x174	201703232232001881						8.13s	MacOSX	6	6 0.00.952380952380952
4	03.23.17 22:39:09	DeviceID: USBVID_046D:0952	PhysicalScreenSize 309x174	201703232232001881						4.07s	Windows_EPP\	6	
5	03.23.17 22:54:39	DeviceID: USBVID_046D:0952	PhysicalScreenSize 309x174	201703232232001881	25	Male	Student	multiple times pe	multiple times p	8.27s	Windows_EPP\	6	
6	03.24.17 11:23:12	DeviceID: USBVID_046D:0952	PhysicalScreenSize 309x174	201703241120352062	31	Male	Student	multiple times pe	multiple times p	6.9s	Logitech	6	
7	03.23.17 22:59:31	DeviceID: USBVID_046D:0952	PhysicalScreenSize 309x174	201703232232001881	25	Male	Student	multiple times pe	multiple times p	6.8s	MacOSX	6	6 0.00.952380952380952
8	03.23.17 23:40:45	DeviceID: USBVID_046D:0952	PhysicalScreenSize 309x174	201703232232001881	25	Male	Student	multiple times pe	multiple times p	6.5s	Windows_EPP	6	
9	03.23.17 23:41:15	DeviceID: USBVID_046D:0952	PhysicalScreenSize 309x174	201703232232001881	25	Male	Student	multiple times pe	multiple times p	4.98s	Windows_EPP	6	6 0.00.952380952380952
10	03.23.17 23:44:14	DeviceID: USBVID_046D:0952	PhysicalScreenSize 309x174	201703232232001881	25	Male	Student	multiple times pe	multiple times p	5.38s	MacOSX	6	
11	03.23.17 23:44:41	DeviceID: USBVID_046D:0952	PhysicalScreenSize 309x174	201703232232001881	25	Male	Student	multiple times pe	multiple times p	15.97s	Windows_EPP	6	6 0.00.952380952380952
12	03.23.17 23:49:04	DeviceID: USBVID_046D:0952	PhysicalScreenSize 309x174	201703232232001881	25	Male	Student	multiple times pe	multiple times p	70.97s	Windows_EPP	6	
13	03.24.17 08:20:5	DeviceID: USBVID_046D:0952	PhysicalScreenSize 309x174	201703232232001881	25	Male	Student	multiple times pe	multiple times p	620.49s	Windows_EPP	6	
14	03.24.17 11:20:50	DeviceID: USBVID_046D:0952	PhysicalScreenSize 309x174	201703241120352062	25	Male	Student	multiple times pe	multiple times p	7.39s	Windows_EPP	6	
15	03.24.17 11:21:09	DeviceID: USBVID_046D:0952	PhysicalScreenSize 309x174	201703241120352062	25	Male	Student	multiple times pe	multiple times p	8.89s	MacOSX	6	6 0.00.952380952380952
16	03.24.17 11:23:35	DeviceID: USBVID_046D:0952	PhysicalScreenSize 309x174	201703241120352062	31	Male	Student	multiple times pe	multiple times p	18s	Logitech	6	6 0.00.952380952380952
17	03/24/17 12:21:28	DeviceID: USBVID_046D:0952	PhysicalScreenSize 235x132	201703241219255402	33	Male	Student	multiple times pe	multiple times p	16.27s	Windows_EPP	6	6 0.00.952380952380952
18	03/24/17 12:22:03	DeviceID: USBVID_046D:0952	PhysicalScreenSize 235x132	201703241219255402	33	Male	Student	multiple times pe	multiple times p	2.36s	Windows_EPP	6	6 0.00.952380952380952
19	03/24/17 12:22:42	DeviceID: USBVID_046D:0952	PhysicalScreenSize 235x132	201703241219255402	33	Male	Student	multiple times pe	multiple times p	13.39s	Windows_EPP	6	
20	03/24/17 12:26:15	DeviceID: USBVID_046D:0952	PhysicalScreenSize 235x132	201703241219255402	33	Male	Student	multiple times pe	multiple times p	25.14s	Windows_EPP	6	6 0.00.952380952380952
21	03.24.17 22:25:35	DeviceID: USBVID_046D:0952	PhysicalScreenSize 309x174	201703241120352062	31	Male	Student	multiple times pe	multiple times p	25.88s	MacOSX	6	
22	03.24.17 22:25:35	DeviceID: USBVID_046D:0952	PhysicalScreenSize 309x174	201703241120352062	31	Male	Student	multiple times pe	multiple times p	13.66s	MacOSX	6	6 0.00.952380952380952
23	03.24.17 22:25:45	DeviceID: USBVID_046D:0952	PhysicalScreenSize 309x174	201703241120352062	31	Male	Student	multiple times pe	multiple times p	3.58s	MacOSX	6	6 0.00.952380952380952
24	03.24.17 22:26:29	DeviceID: USBVID_046D:0952	PhysicalScreenSize 309x174	201703241120352062	31	Male	Student	multiple times pe	multiple times p	13.55s	MacOSX	6	6 0.00.952380952380952
25	03.24.17 22:27:26	DeviceID: USBVID_046D:0952	PhysicalScreenSize 309x174	201703241120352062	31	Male	Student	multiple times pe	multiple times p	41.99s	MacOSX	6	6 0.00.952380952380952
26	03/25/17 15:41:24	DeviceID: USBVID_046D:0952	PhysicalScreenSize 309x174	201703251539544417	27	Male	Student	multiple times pe	multiple times p	10.08s	Windows_EPP	6	6 0.00.957208541052707
27	03/25/17 15:41:43	DeviceID: USBVID_046D:0952	PhysicalScreenSize 309x174	201703251539544417	27	Male	Student	multiple times pe	multiple times p	13.69s	Windows_EPP	6	6 0.00.957208541052707
28	03.25.17 22:10:49	DeviceID: ACPI\LL04	PhysicalScreenSize 309x174	201703252203531593						324.2s	Windows_EPP	6	
29	03.26.17 1:48:12	DeviceID: ACPI\LL04	PhysicalScreenSize 309x174	201703262203531593	32	Male	Student	multiple times pe	multiple times p	5.6s	Windows_EPP	6	6 0.00.952380952380952
30	03.26.17 1:48:40	DeviceID: ACPI\LL04	PhysicalScreenSize 309x174	201703262203531593	32	Male	Student	multiple times pe	multiple times p	10.56s	MacOSX	6	6 0.00.952380952380952
31		CustomizedTransferFunctions											

Figure A.6: Screen short of Google sheets - customized transfer functions.

	A	B	C	D	E	F	G	H	I	J
1	timestamp	PointingDevicesInfos	DisplayDevicesInfos	userID	age	sex	profession	useHouse	useTouchpad	Questions
41	03.20.17.4.06.42	DeviceID: USBVID_046D&PhysicalScreenSize	309x1201703182200266885	30	Male	Student	multiple times pt	multiple times pt	Windows_EPP 6 too slow, Windows_EPP 6 fits r	
42	03.20.17.4.24.36	DeviceID: USBVID_046D&PhysicalScreenSize	309x1201703182200266885	30	Male	Student	multiple times pt	multiple times pt	MacOSX 6 too slow, MacOSX 6 fits my needs, MacOSX 6	
43	03.20.17.14.32.23	DeviceID: USBVID_093A&PhysicalScreenSize	518x3201703201429151693	30	Male	Employee	multiple times pt	not at all	Windows_NoEPP 1 too fast, Windows_NoEPP 1 too slow, Windows_NoEPP	
44	03.20.17.14.45.41	DeviceID: HIDVID_05AC&PhysicalScreenSize	286x1201703201405058992	36	Male	Professor	multiple times pt	multiple times pt	Windows_EPP 3 too slow, Windows_EPP 3 fits r	
45	03.20.17.14.47.11	DeviceID: HIDVID_05AC&PhysicalScreenSize	286x1201703201405058992	36	Male	Professor	multiple times pt	multiple times pt	Windows_EPP 3 too fast, Windows_EPP 3 too slow, Windows_EPP 3 fits r	
46	03.20.17.14.48.11	DeviceID: HIDVID_05AC&PhysicalScreenSize	286x1201703201405058992	36	Male	Professor	multiple times pt	multiple times pt	Windows_EPP 3 too fast, Windows_EPP 3 too slow, Windows_EPP 3 fits r	
47	03.20.17.21.59.12	DeviceID: USBVID_046D&PhysicalScreenSize	309x1201703182200266885	30	Male	Student	multiple times pt	multiple times pt	Windows_EPP 7 too slow, Windows_EPP 7 fits r	
48	03.21.17.0.48.22	DeviceID: USBVID_046D&PhysicalScreenSize	309x1201703182200266885	30	Male	Student	multiple times pt	multiple times pt	Windows_EPP 6 too fast, Windows_EPP 6 too slow, Windows_EPP 6 fits r	
49	03.21.17.0.58.04	DeviceID: USBVID_046D&PhysicalScreenSize	309x1201703182200266885	30	Male	Student	multiple times pt	multiple times pt	Windows_EPP 6 too fast, Logitech 6 too slow, Logitech 6 fits my needs, Logitech	
50	03.21.17.1.14.12	DeviceID: USBVID_046D&PhysicalScreenSize	309x1201703182200266885	30	Male	Student	multiple times pt	multiple times pt	Windows_EPP 6 too fast, Logitech 6 too slow, Logitech 6 fits my needs, Logitech	
51	03.21.17.21.24.18	DeviceID: USBVID_046D&PhysicalScreenSize	309x1201703212121309591	30	Male	Rupak	multiple times pt	multiple times pt	Windows_EPP 6 too fast, Windows_EPP 6 too slow, Windows_EPP 6 fits r	
52	03.21.17.21.26.36	DeviceID: USBVID_046D&PhysicalScreenSize	309x1201703212121309591	30	Male	Rupak	multiple times pt	multiple times pt	MacOSX 6 too fast, MacOSX 6 too slow, MacOSX 6 fits my needs, MacOSX 6	
53	03.21.17.22.16.30	DeviceID: ACPIETD063014	PhysicalScreenSize 344x1201703212211082359	25	Male	Student	multiple times pt	multiple times pt	Windows_EPP 6 too fast, Windows_EPP 6 too slow, Windows_EPP 6 fits r	
54	03.23.17.10.02.44	DeviceID: USBVID_046D&PhysicalScreenSize	309x1201703182200266885	30	Male	Student	multiple times pt	multiple times pt	Windows_EPP 6 too fast, Logitech 11 too slow, Logitech 11 fits my needs, Logith	
55	03.23.17.10.56.34	DeviceID: USBVID_046D&PhysicalScreenSize	309x1201703182200266885	30	Male	Student	multiple times pt	multiple times pt	Windows_EPP 6 too fast, Windows_EPP 6 too slow, Windows_EPP 6 fits r	
56	03.23.17.12.35.25	DeviceID: USBVID_093A&PhysicalScreenSize	518x3201703201429151693	30	Male	Employee	multiple times pt	not at all	Windows_NoEPP 11 too fast, Windows_NoEPP 11 too slow, Windows_NoEPP	
57	03.23.17.12.37.54	DeviceID: USBVID_093A&PhysicalScreenSize	518x3201703201429151693	30	Male	Employee	multiple times pt	not at all	Windows_NoEPP 9 too fast, Windows_NoEPP 9 too slow, Windows_NoEPP	
58	03.23.17.14.09.21	DeviceID: HIDVID_05AC&PhysicalScreenSize	286x1201703231408262294	25	Male	Student	multiple times pt	multiple times pt	Windows_EPP 6 too fast, Windows_EPP 6 too slow, Windows_EPP 6 fits r	
59	03.23.17.14.31.41	DeviceID: HIDVID_05AC&PhysicalScreenSize	286x1201703231408262294	25	Male	Student	multiple times pt	multiple times pt	Windows_EPP 6 too fast, Windows_EPP 6 too slow, Windows_EPP 6 fits r	
60	03.23.17.18.42.16	DeviceID: USBVID_046D&PhysicalScreenSize	309x1201703182200266885	30	Male	Student	multiple times pt	multiple times pt	Windows_EPP 6 too fast, Windows_EPP 6 too slow, Windows_EPP 6 fits r	
61	03.23.17.21.12.16	DeviceID: USBVID_046D&PhysicalScreenSize	309x1201703182200266885	30	Male	Student	multiple times pt	multiple times pt	MacOSX 6 too fast, MacOSX 6 too slow, MacOSX 6 fits my needs, MacOSX 6	
62	03.23.17.21.13.05	DeviceID: USBVID_046D&PhysicalScreenSize	309x1201703182200266885	30	Male	Student	multiple times pt	multiple times pt	Logitech 6 too fast, Logitech 6 fits my needs, Logitech	
63	03.23.17.22.40.23	DeviceID: USBVID_046D&PhysicalScreenSize	309x120170322232001881	25	Male	Student	multiple times pt	multiple times pt	Windows_EPP 6 too fast, Windows_EPP 6 too slow, Windows_EPP 6 fits r	
64	03.23.17.22.54.56	DeviceID: USBVID_046D&PhysicalScreenSize	309x120170322232001881	25	Male	Student	multiple times pt	multiple times pt	Windows_EPP 6 too fast, Windows_EPP 6 too slow, Windows_EPP 6 fits r	
65	03.23.17.22.59.45	DeviceID: USBVID_046D&PhysicalScreenSize	309x120170322232001881	25	Male	Student	multiple times pt	multiple times pt	MacOSX 6 too fast, MacOSX 6 too slow, MacOSX 6 fits my needs, MacOSX 6	
66	03.23.17.23.45.07	DeviceID: USBVID_046D&PhysicalScreenSize	309x120170322232001881	25	Male	Student	multiple times pt	multiple times pt	Windows_EPP 6 too fast, Windows_EPP 6 too slow, Windows_EPP 6 fits r	
67	03.24.17.0.08.41	DeviceID: USBVID_046D&PhysicalScreenSize	309x120170322232001881	25	Male	Student	multiple times pt	multiple times pt	Windows_EPP 6 too fast, Windows_EPP 6 too slow, Windows_EPP 6 fits r	
68	03.24.17.11.21.54	DeviceID: USBVID_046D&PhysicalScreenSize	309x1201703241219255402	33	Male	Student	multiple times pt	multiple times pt	MacOSX 6 too fast, MacOSX 6 too slow, MacOSX 6 fits my needs, MacOSX 6	
69	03.24.17.12.23.18	DeviceID: USBVID_24AE&PhysicalScreenSize	235x1201703241219255402	33	Male	Student	multiple times pt	multiple times pt	Windows_EPP 6 too fast, Windows_EPP 6 too slow, Windows_EPP 6 fits r	
70	03.24.17.12.26.34	DeviceID: USBVID_24AE&PhysicalScreenSize	235x1201703241219255402	33	Male	Student	multiple times pt	multiple times pt	Windows_EPP 9 too fast, Windows_EPP 9 too slow, Windows_EPP 9 fits r	
71	03.24.17.12.31.44	DeviceID: USBVID_24AE&PhysicalScreenSize	235x1201703241219255402	33	Male	Student	multiple times pt	multiple times pt	Windows_EPP 9 too fast, Windows_EPP 9 too slow, Windows_EPP 9 fits r	
72	03.24.17.12.44.08	DeviceID: USBVID_24AE&PhysicalScreenSize	235x1201703241219255402	33	Male	Student	multiple times pt	multiple times pt	Windows_EPP 9 too fast, Windows_EPP 9 too slow, Windows_EPP 9 fits r	
73	03.25.17.15.42.06	DeviceID: USBVID_0461&PhysicalScreenSize	309x1201703251539544417	27	Male	Student	multiple times pt	multiple times pt	Windows_EPP 6 too fast, Windows_EPP 6 too slow, Windows_EPP 6 fits r	
14	⏪ ⏩ 🏠 📄 🗑️	UserResponsesFromDesktop	UserQuans							

Figure A.7: Screen short of Google sheets - User defined transfer function feed-back[part 1].

A.4 libpointing data points

A. Appendix

Motor Speed(m/s)	Visual Speed(m/s)	Motor Speed(m/s)	Visual Speed(m/s)	Motor Speed(m/s)	Visual Speed(m/s)
0	0	43	96.67	85	212.22
1	0.58	44	99.43	86	214.97
2	1.31	45	102.18	87	217.72
3	2.18	46	104.93	88	220.47
4	3.07	47	107.68	89	223.22
5	4.22	48	110.44	90	225.97
6	5.56	49	113.18	91	228.73
7	6.88	50	115.93	92	231.47
8	8.22	51	118.69	93	234.23
9	9.55	52	121.43	94	236.97
10	10.88	53	124.19	95	239.73
11	12.21	54	126.94	96	242.48
12	13.54	55	129.69	97	245.23
13	14.87	56	132.44	98	247.98
14	16.9	57	135.19	99	250.73
15	19.65	58	137.94	100	253.48
16	22.4	59	140.69	101	256.23
17	25.16	60	143.45	102	258.99
18	27.9	61	146.19	103	261.73
19	30.65	62	148.95	104	264.49
20	33.41	63	151.7	105	267.23
21	36.16	64	154.44	106	269.99
22	38.9	65	157.2	107	272.74
23	41.66	66	159.95	108	275.49
24	44.41	67	162.7	109	278.24
25	47.16	68	165.45	110	280.99
26	49.91	69	168.2	111	283.74
27	52.67	70	170.96	112	286.5
28	55.41	71	173.7	113	289.24
29	58.16	72	176.46	114	292
30	60.92	73	179.2	115	294.74
31	63.66	74	181.96	116	297.5
32	66.42	75	184.71	117	300.25
33	69.17	76	187.46	118	303
34	71.92	77	190.21	119	305.75
35	74.67	78	192.96	120	308.5
36	77.42	79	195.71	121	311.25
37	80.17	80	198.46	122	314
38	82.92	81	201.22	123	316.76
39	85.68	82	203.96	124	319.5
40	88.42	83	206.72	125	322.26
41	91.18	84	209.46	126	325
42	93.93			127	327.76

Table A.1: Windows Transfer Function data points set when EPP on and Speed 6.

A.5 Trials and Feedbacks Activated Duration.

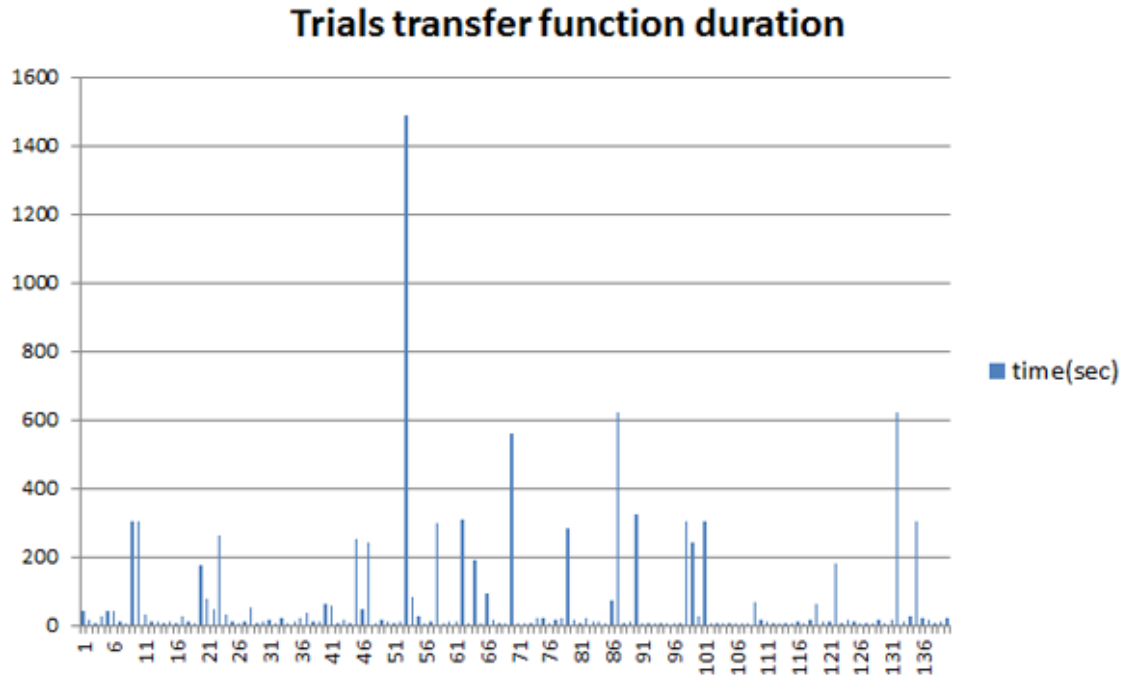


Figure A.9: Trails transfer function activated time.

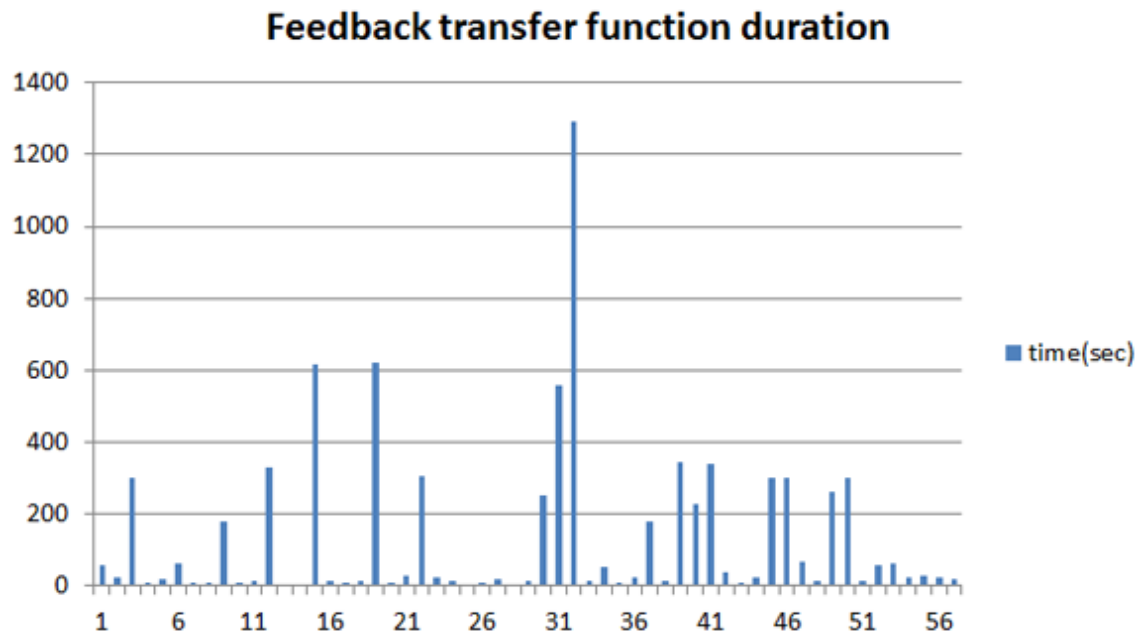


Figure A.10: Feedback transfer function activated time.

A.6 Interesting Transfer Function for Individual Participants.

A.6. Interesting Transfer Function for Individual Participants.

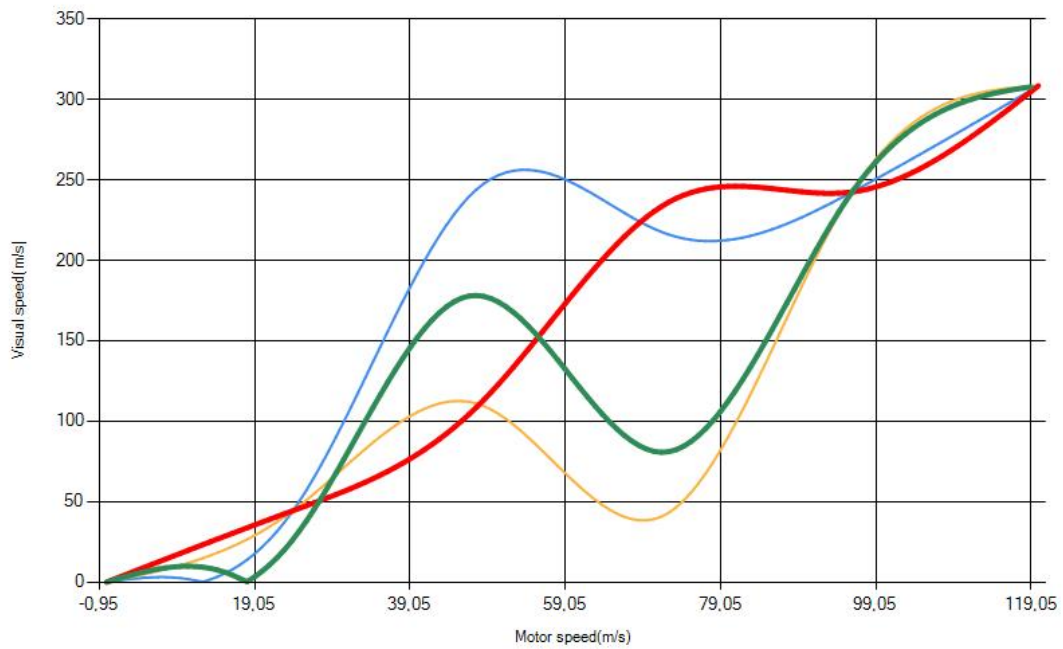


Figure A.11: Interesting Transfer Function for Participants User9(best fits marked green and longest activated trial marked red).

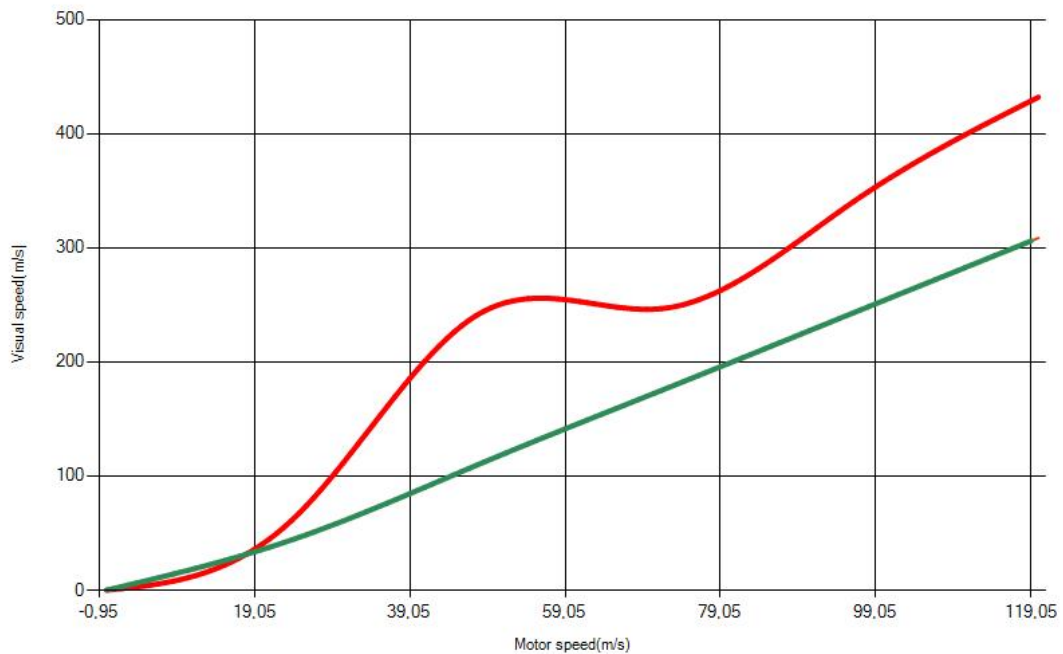


Figure A.12: Interesting Transfer Function for Participants User10(best fits marked green and longest activated trial marked red).

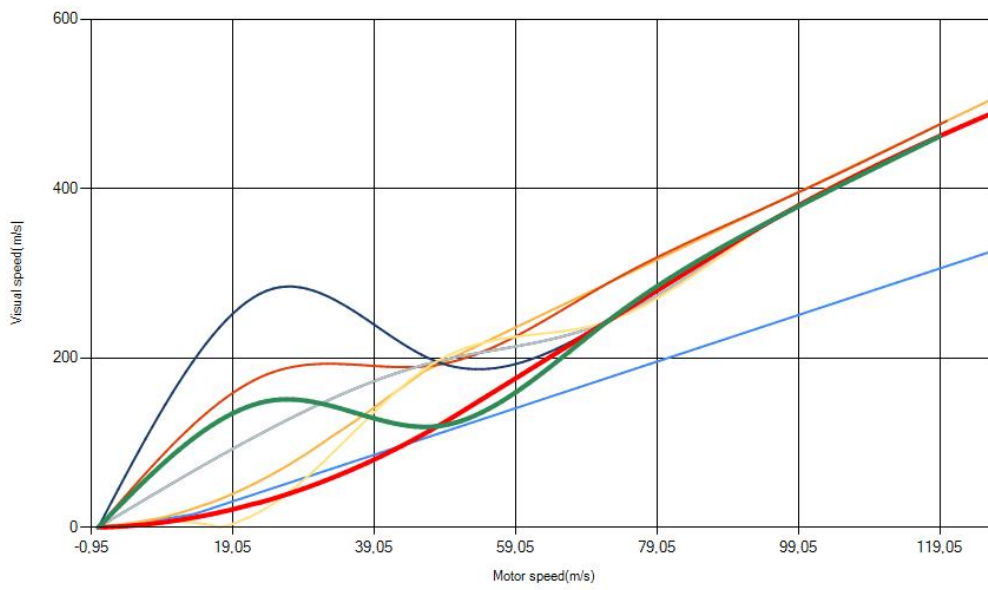


Figure A.13: Interesting Transfer Function for Participants User11(best fits marked green and longest activated trial marked red).

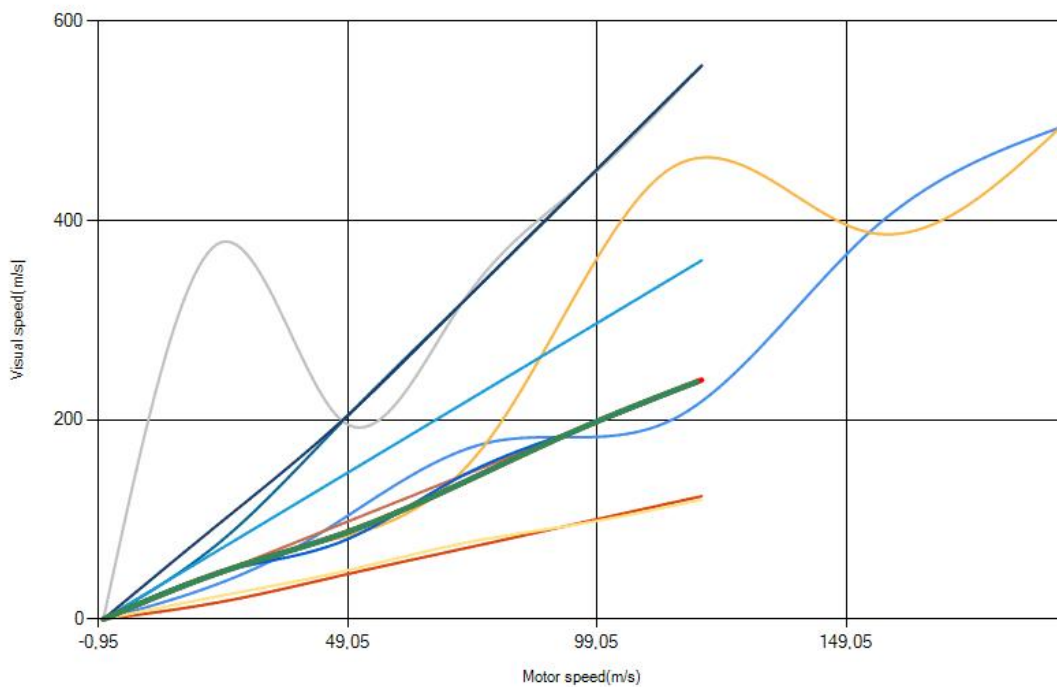


Figure A.14: Interesting Transfer Function for Participants User12(best fits marked green and longest activated trial marked red).

A.6. Interesting Transfer Function for Individual Participants.

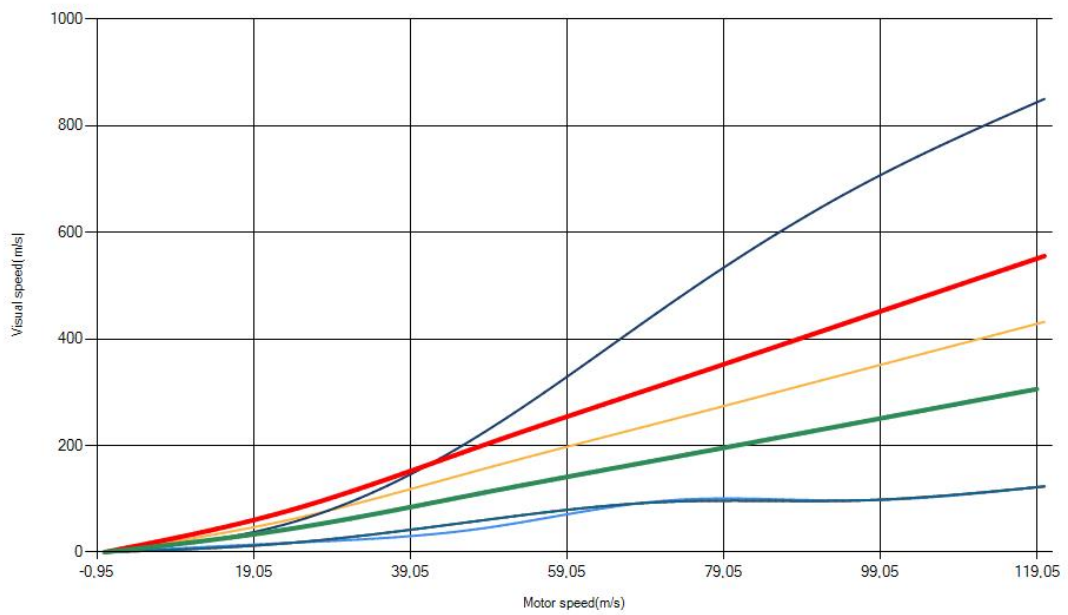


Figure A.15: Interesting Transfer Function for Participants User15(best fits marked green and longest activated trial marked red).

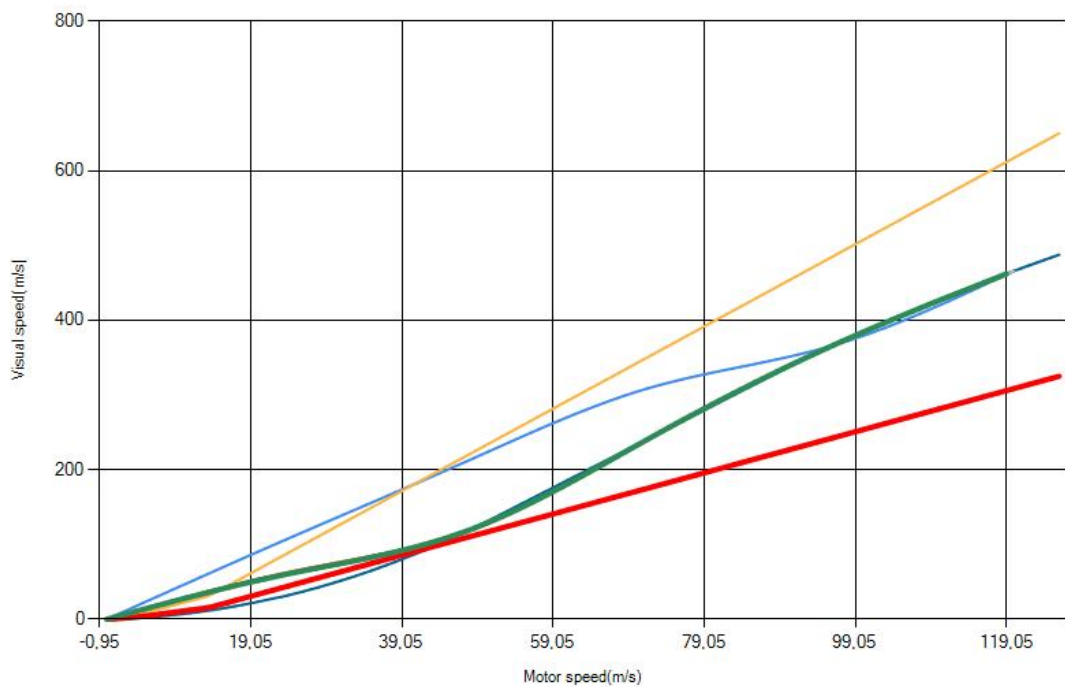


Figure A.16: Interesting Transfer Function for Participants User19(best fits marked green and longest activated trial marked red).

A. Appendix

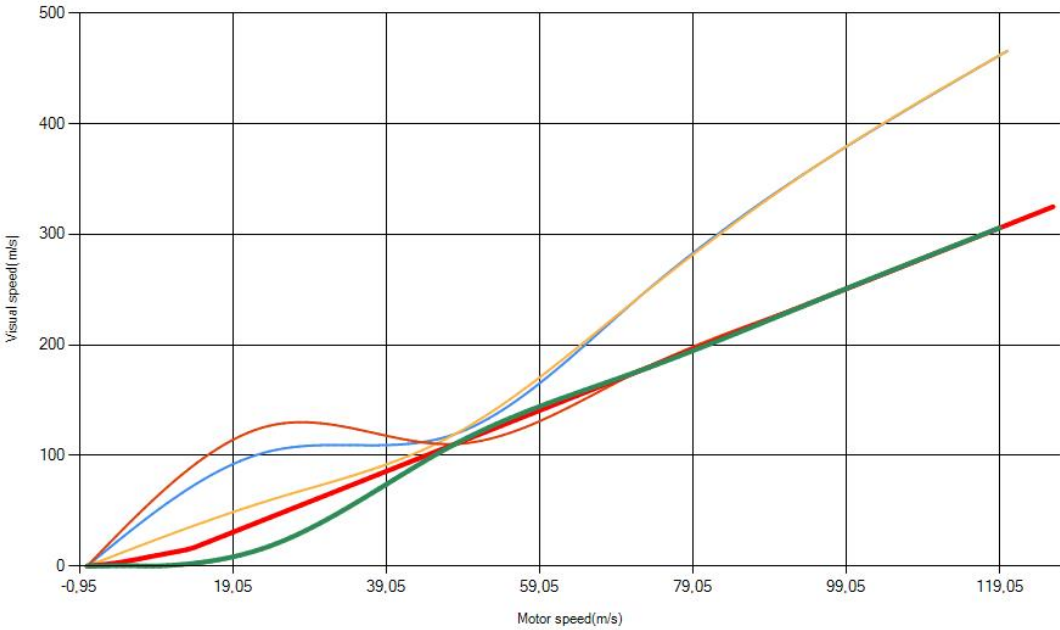


Figure A.17: Interesting Transfer Function for Participants User20(best fits marked green and longest activated trial marked red).

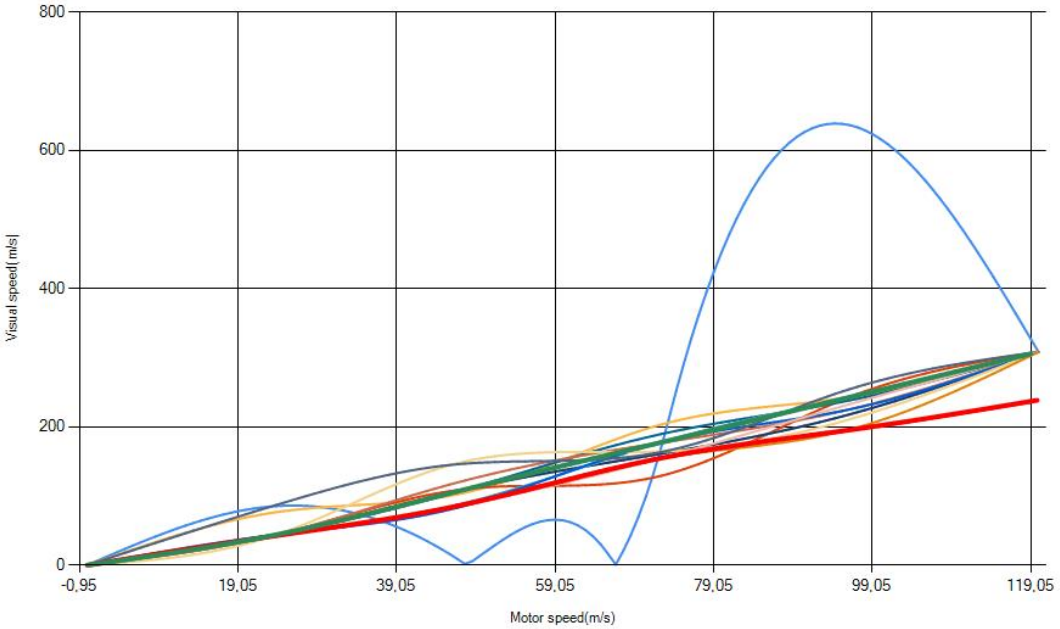


Figure A.18: Interesting Transfer Function for Participants User21(best fits marked green and longest activated trial marked red).

A.6. Interesting Transfer Function for Individual Participants.

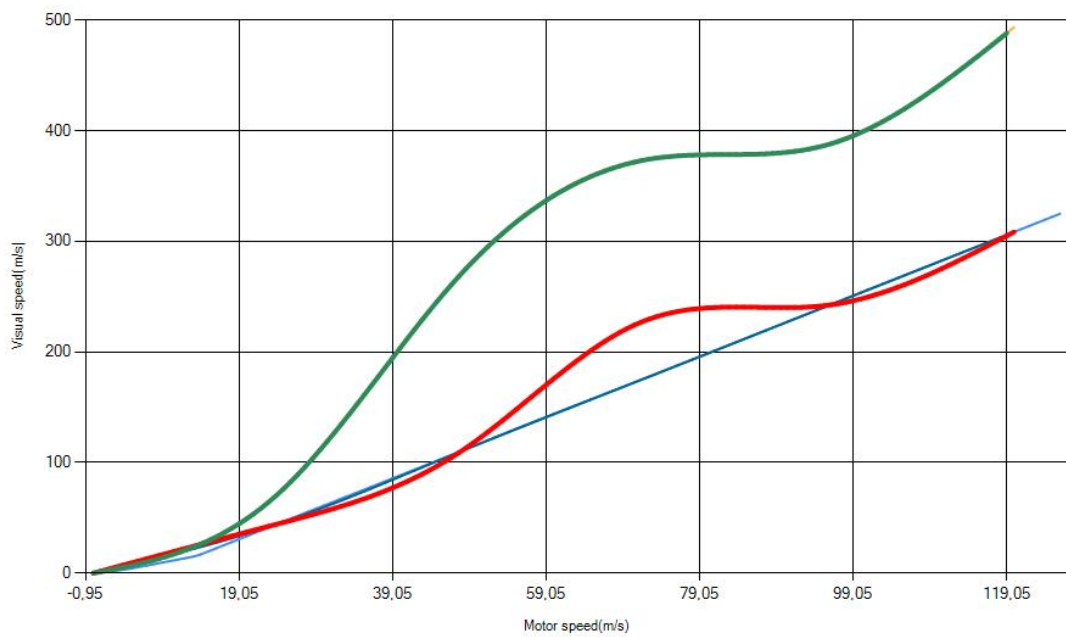


Figure A.19: Interesting Transfer Function for Participants User22(best fits marked green and longest activated trial marked red).

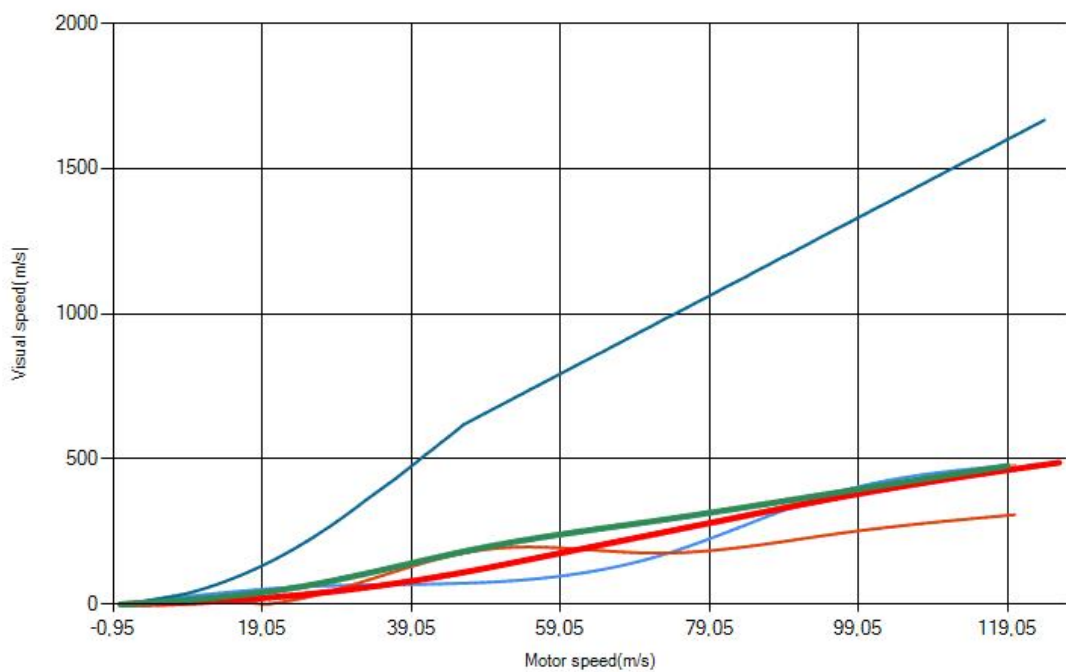


Figure A.20: Interesting Transfer Function for Participants User23(best fits marked green and longest activated trial marked red).

A.7 Box Plot Diagram.

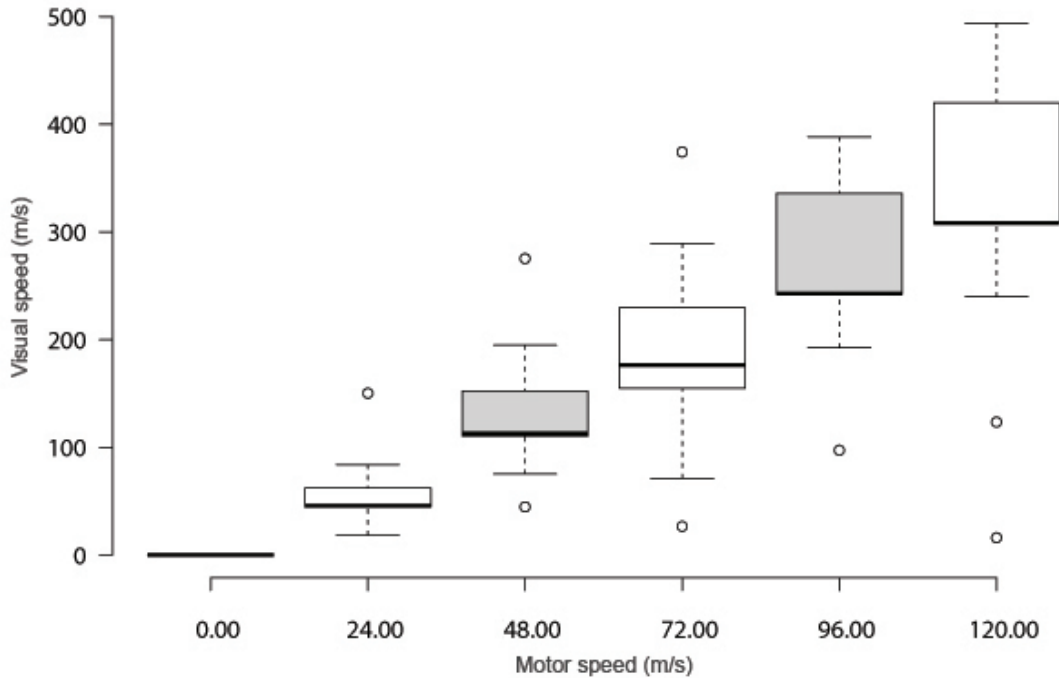


Figure A.21: Box plot graph for mouse interesting transfer function based on agreed feedback response.

	0.00	24.00	48.00	72.00	96.00	120.00
Upper whisker	0.00	84.00	194.90	289.41	388.36	494.00
3rd quartile	0.00	62.24	152.17	229.56	336.00	420.00
Median	0.00	45.80	113.14	176.46	243.01	308.50
1st quartile	0.00	44.41	110.44	154.83	242.48	308.50
Lower whisker	0.00	18.49	75.22	71.00	192.41	240.00
Nr. of data points	29.00	29.00	29.00	29.00	29.00	29.00

Table A.2: Box plot diagram statistics for the mouse interesting transfer function based on agreed feedback response.

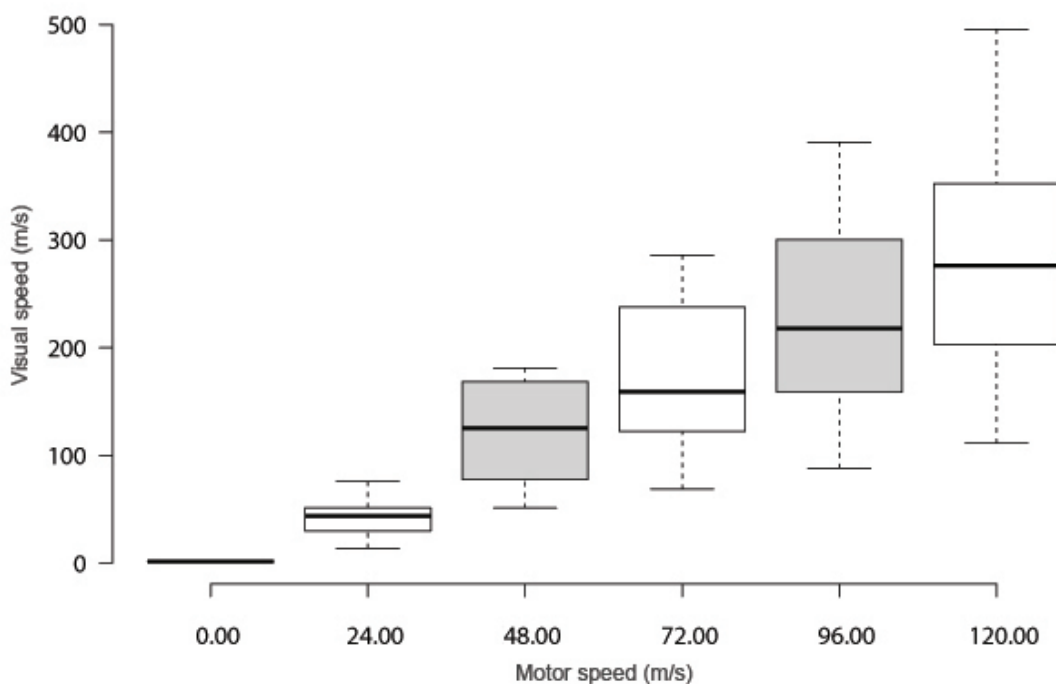


Figure A.22: Box plot graph for touchpad interesting transfer function based on agreed feedback response.

	0.00	24.00	48.00	72.00	96.00	120.00
Upper whisker	0.00	83.16	201.72	319.49	437.41	555.30
3rd quartile	0.00	55.97	187.65	265.18	335.72	394.25
Median	0.00	47.15	138.98	177.06	243.07	308.50
1st quartile	0.00	31.57	85.42	135.63	176.59	226.59
Lower whisker	0.00	13.71	55.71	75.30	97.07	123.40
Nr. of data points	8.00	8.00	8.00	8.00	8.00	8.00

Table A.3: Box plot diagram statistics for the touchpad interesting transfer function based on agreed feedback response.

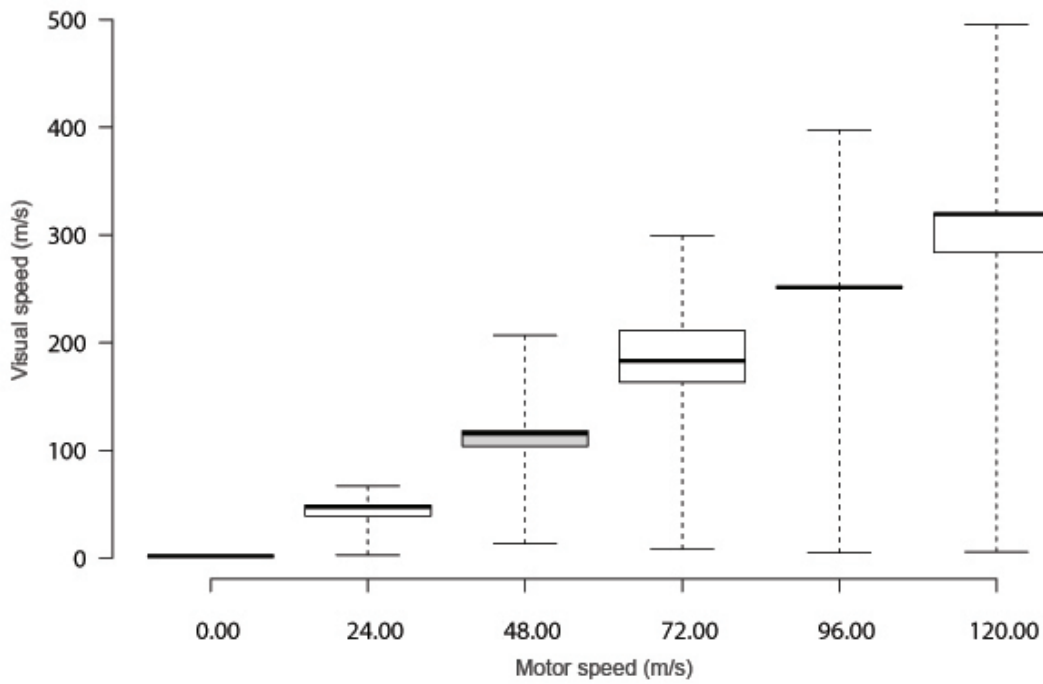


Figure A.23: Box plot graph for mouse interesting transfer function based on trial activated time.

	0.00	24.00	48.00	72.00	96.00	120.00
Upper whisker	0.00	49.29	113.23	244.27	243.01	308.50
3rd quartile	0.00	46.00	113.18	203.65	242.99	308.50
Median	0.00	44.41	110.44	176.46	242.48	308.50
1st quartile	0.00	36.42	99.32	157.01	242.48	274.25
Lower whisker	0.00	28.14	86.83	137.56	242.48	240.00
Nr. of data points	15.00	15.00	15.00	15.00	15.00	15.00

Table A.4: Box plot statistics for mouse interesting transfer function based on trial activated time.

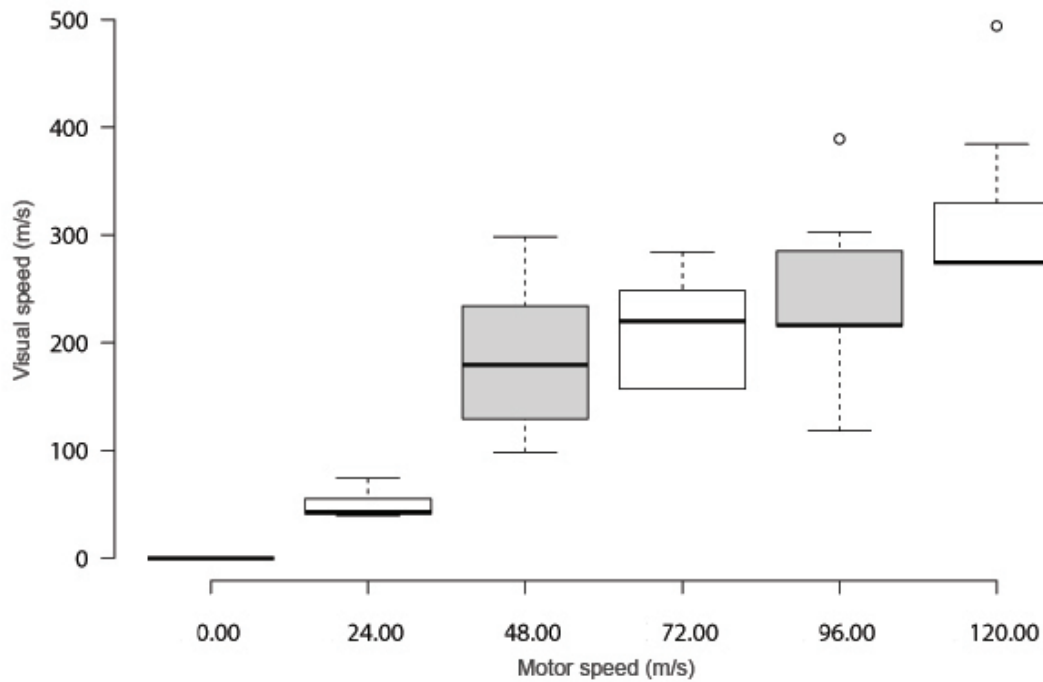


Figure A.24: Box plot diagram statistics for the mouse interesting transfer function based on trial activated time.

	0.00	24.00	48.00	72.00	96.00	120.00
Upper whisker	0.00	83.16	335.20	319.49	340.35	431.90
3rd quartile	0.00	62.00	262.98	278.92	320.28	370.20
Median	0.00	48.10	201.72	247.41	243.13	308.50
1st quartile	0.00	46.07	145.36	176.54	242.48	308.50
Lower whisker	0.00	44.41	110.44	176.46	132.78	308.50
Nr. of data points	7.00	7.00	7.00	7.00	7.00	7.00

Table A.5: Box plot diagram statistics for the touchpad interesting transfer function based on trial activated time.

A.8 Interesting Transfer Functions.

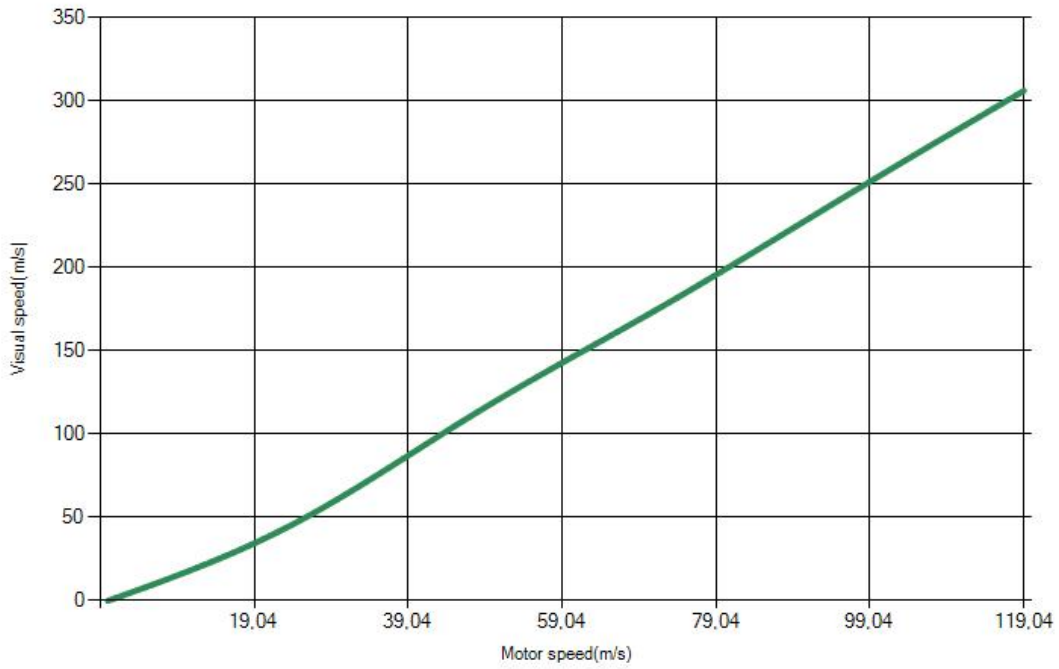


Figure A.25: Interesting transfer function for mouse based on agreed feedback response.

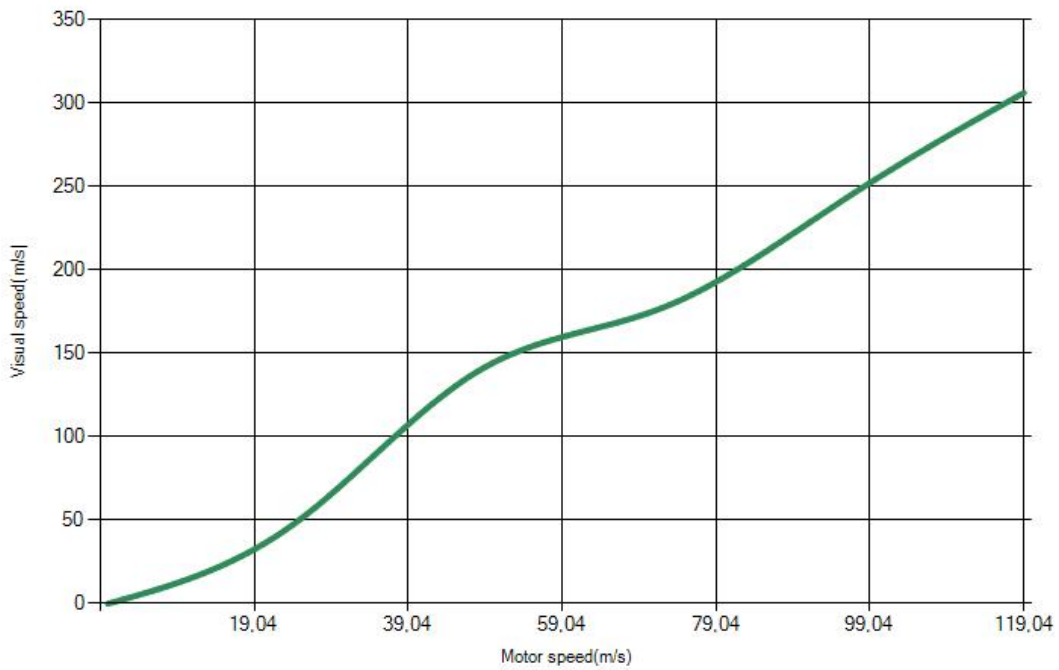


Figure A.26: Interesting transfer function for touchpad based on agreed feedback response.

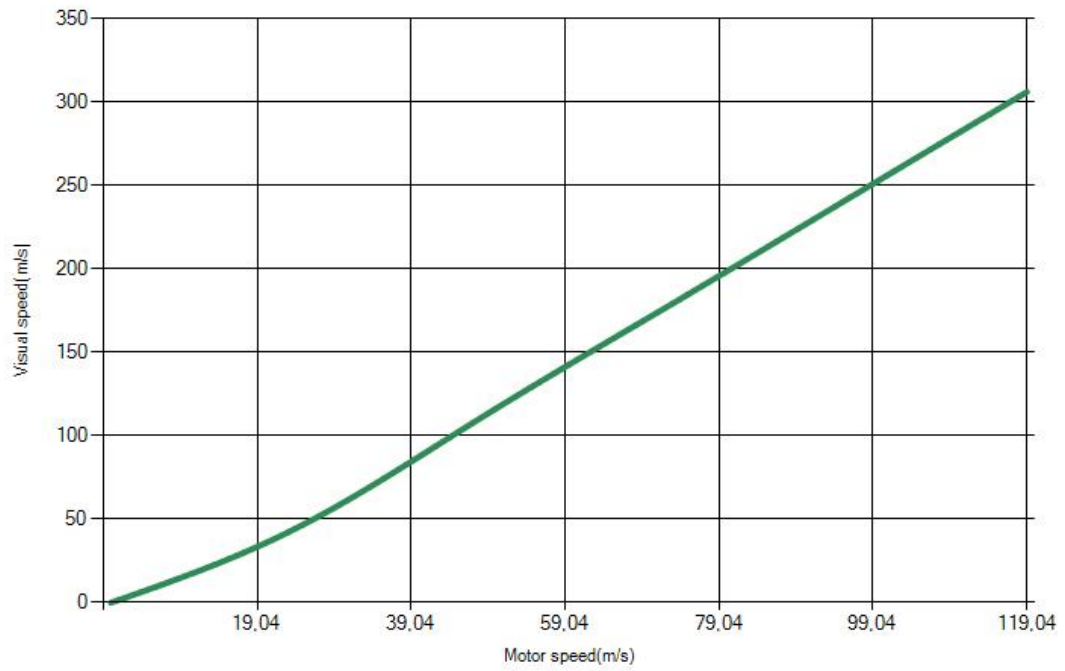


Figure A.27: Interesting transfer function for mouse based on trial activated time.

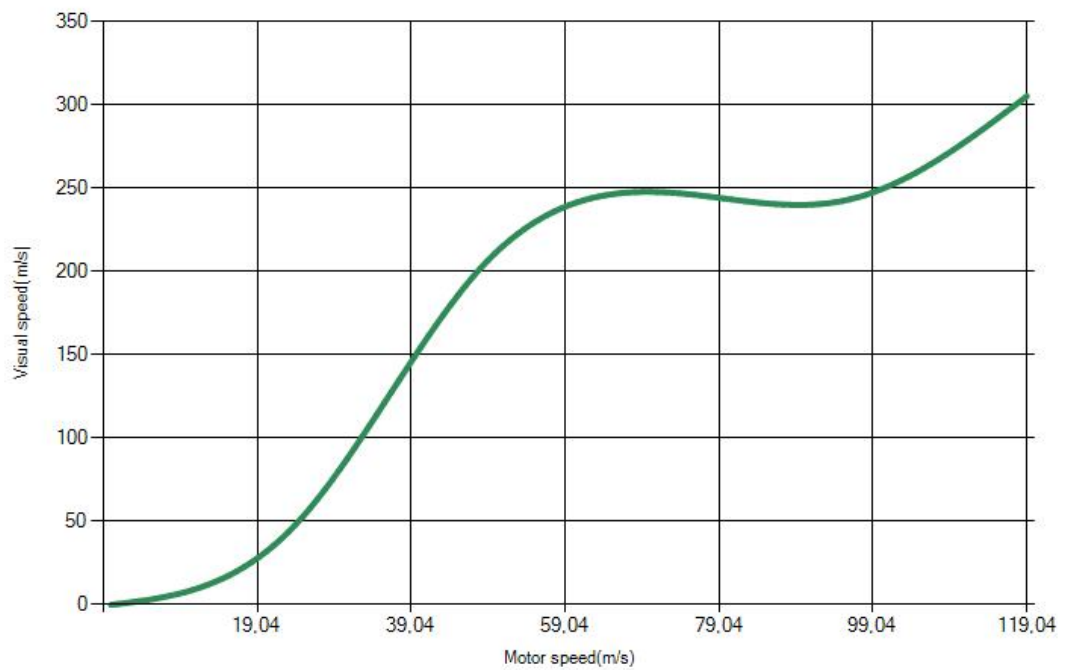


Figure A.28: Interesting transfer function for touchpad based on trial activated time.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

Stuttgart, 12.05.2017

(S.M. Hasanul Banna)