

B.DES PROJECT 1

Single-Handed Input for Numeric Data

Atish Waghware | 18U130008

Supervisor: Prof. Jayesh Pillai



Declaration

I declare that this written document represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/ source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

A handwritten signature in black ink, appearing to read 'Atish', with a large, stylized flourish extending to the right.

Atish Waghwase

18U130008

IDC School of Design, IIT Bombay.

25th November 2018

Approval

The B.Des Project 1 entitled “Single-Handed Input for Numeric Data” by Atish Waghware, Roll Number 18U130008 is approved, in fulfillment of the Bachelor’s in Design Degree at IDC School of Design, Indian Institute of Technology Bombay.

Internal:

External:

Guide:

Chairperson:

Acknowledgement

I would firstly like to thank my supervisor Prof. Jayesh Pillai for his constant guidance and support and for giving me knowledge and exposure in the field of interactive media. I also want to thank my jury - Prof. Venkatesh Rajamanikam, Prof. Vivek Kant and Prof. Swati Pal - for their valuable feedback. Special thanks to Prof. Anirudha Joshi for all the valuable knowledge he imparted to me and for going out of his way to help this project. Lastly, thanks to my family and friends at IITB for their support without which this project would not have been possible.

Abstract

This project attempts to explore new interaction techniques in the field of human-computer interaction (HCI). The initial area of focus was developing new interactions for CAD applications. CAD workflows of users as well as the current technologies, controllers and interaction methods were studied. I ideated on potential solutions to problems or novel interactions to enhance these workflows. However, Fingerspelling Numeric Input as a concept was picked as the redefined design brief based on scope and novelty. Two prototypes were made to simulate two possible physical implementations. Evaluation shows that it is easy to learn fingerspelling gestures and they compare quite well to conventional devices like numpads in certain scenarios. I concluded that fingerspelling interactions can act as a viable means of single-handed input, provided the encoding scheme and ergonomics are optimal.

Contents

Declaration	1	2.6 Problem Identification	12
Approval	2	Ideation and Conception	13
Acknowledgement	3	3.1 Ideation	14
Abstract	4	3.2 Idea Evaluation	14
Contents	5	3.3 Restated Design Brief	15
Introduction	8	3.4 Concepts	16
1.1 Introduction	8	3.5 Chosen Concept	19
1.2 Motivation	8	Design and Detailing	19
1.3 Scope and Limitations	8	4.1 Description	19
1.4 Methodology	9	4.2 Gestures	20
Initial Study	9	4.3 Working Mechanism	20
2.1 Background	9	Evaluation	23
2.2 Initial focus	9	5.1 Apparatus	23
2.3 User Study	9	5.2 Limitations	23
2.4 Secondary Research	10	5.3 Methodology	24
2.5 Related Work	11	5.4 Demographics	24
		5.5 Metrics	25

5.6 Test 1	25
5.7 Test 2	27
5.7 Difficulty Rating Survey	29
Results	29
6.1 Test 1 Results	29
6.2 Test 2 Results	30
6.3 Difficulty Rating	31
6.4 Interview Findings	31
Discussion	34
7.1.1 Learning Curve	34
7.1.2 Gestures and Encoding	34
7.1.3 Ergonomics	35
7.1.4 Test 2	36
Conclusion	36
8.1 Conclusion	36
8.2 Additional Explorations	37
8.3 Future Scope	38
References	39

Introduction

1.1 Introduction

Computer Aided Design or CAD is a way to design 2D or 3D objects in virtual environments before they are manufactured or made, and is a fundamental skill that designers and engineers in the 21st century need to have. There is a plethora of CAD tools available to anyone who wants to utilize them, and each tool has its own set of quirks. An aspect they all share, however, is the fact that all virtual entities are located in a virtual environment and the user has to navigate that environment in order to modify those entities. The thing about navigating these environments is that a lot of tools provide 2D representations of 3D environments and the controls being used to navigate them are also not meant for 3D navigation. This project attempts to explore these controls and find better ways to control virtual objects.

1.2 Motivation

As a design student, I have had to work extensively with CAD software for my academic projects and I have noticed these quirks over time, one of which was object exploration using a keyboard and mouse. I found that I often had to zigzag my way through the digital space to get the view I wanted. I decided that this was highly inefficient and quite unnatural.

At the same time, I tried out apps like Quill and Gravity Sketch in virtual reality, the controls for which felt much more natural. Not to say that VR interaction principles can be applied to a 2D CAD environment, but this experience led me to dive deeper into the inconveniences of using CAD software, and with that started my project of designing new interaction methods for 2D CAD workflows.

1.3 Scope and Limitations

The scope of this project is to suggest new interactions and the subsequent development of a novel controller or interaction technique useful mainly in a desktop 3D CAD environment such as Blender, Fusion 360 or Solidworks; basically to generate a new interaction technique and

evaluate it. This project will suggest real life or commercial applications, however the design of said applications isn't in the scope of this project.

1.4 Methodology

An experimental mode of design will be followed where instead of making minor improvements to an existing interaction technique, focus will be on creating a novel interaction technique and evaluating its usability. The problem statement would be in the form of 'what if such an interaction was possible?' Depending on the usability of the resulting technique, subsequent steps or applications can be decided, giving scope for new theories and development.

Initial Study

2.1 Background

Object navigation in CAD environments was the initial area of focus. Object navigation can be defined as the act of moving a virtual object around using pan, rotate or

translate to understand its structure through its on-screen 2D representation. Desktop CAD applications currently require a keyboard and mouse to navigate the CAD environment, which is not optimal since those devices were not built to navigate 3D environments. However, there are special controllers or mice in the market that solve some of the problems of keyboard and mouse navigation (discussed further) but there is still scope for design in this domain.

2.2 Initial focus

I wanted to study how people interact with virtual objects using physical controllers, what the advantages and limitations of current controllers like mice are, and how the abilities of those controllers enhance or hinder object learning and exploration; so initially the focus was set on understanding how people interact with virtual objects using tangible objects (controllers).

2.3 User Study

8 students from IDC School of Design who have multiple years of experience using desktop CAD software such as Fusion 360, Blender, Solidworks and Keyshot were asked

to demonstrate their workflow by making small modifications to their own 3D models. They were interviewed about their habits, annoyances and hindrances in their workflow and took suggestions on what they think could be improved. The way of navigating the 3D environment was specifically observed as that was the initial focus.

2.4 Secondary Research

I tried out various controllers and interaction methods of interacting with virtual 3D objects and used the internet to study those in detail. Being a long term user of CAD software, I did a retrospective heuristic evaluation of my experience using keyboard and mouse for object navigation. The following is a summary of the study -

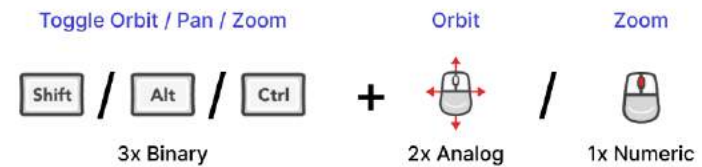


Figure 2.4.1 : Navigation using activator keys

Desktop CAD applications use a keyboard and mouse to let the user navigate their virtual environments; but the mouse is a two dimensional pointing device built for two dimensional navigation only, and it achieves it through analog input in 2 degrees of freedom - Analog input means a device can identify change in its state over a range of values, unlike Digital input which can only know if something has changed between on or off.



Figure 2.4.2 : Types of Input

For example, a key on a typical keyboard provides digital input - a key is either pressed or not pressed; there is no in between. But a mouse (providing analog input) not only knows that it has been moved, but also knows how much it has been moved. It knows how much it has moved along the X-axis and how much along the Y-axis. This means it has 2 degrees of freedom or DOF - one in the X-axis and one in the Y-axis.

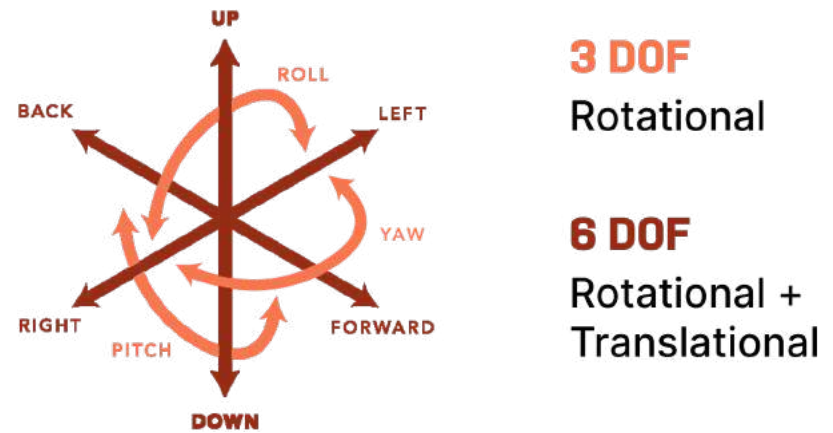


Figure 2.4.3 : Degrees of Freedom

Natural movement in a 3D environment requires 6 degrees of freedom - rotation in the X, Y and Z axes and translation along X, Y and Z axes. Virtual reality headsets have controllers that provide such 6DOF input. The way the mouse currently achieves 6 DOF control is by switching between axes using activator keys; the user can press and hold Key A to activate rotation of the camera in the X and Y axes and utilize the two degrees of freedom, or they can hold Key B to activate pan and translate the camera in the X or Y direction, and so on. For translating in the Z direction, the user has to switch views so that what was initially the Z direction is now X or Y and then translate, or they can use the Zoom function to translate the camera using the scroll wheel which is a pseudo-analog input or numeric input.

Then there are navigation modes which have different control mappings. A turntable navigation mode will feel like the CAD object is placed on a turntable, and dragging the mouse is dragging a turntable. Fly/walk mode is like walking around in a CAD environment, and instead of revolving the camera around the object like in the turntable mode, dragging the mouse will let the user 'look around' the virtual environment. A user often needs constrained movement along only one axis and the mouse does that quite well, but simply exploring an object requires the user to zigzag between axes and switch between pan and zoom. All these methods are workarounds to the problem that the mouse doesn't have enough degrees of freedom to naturally navigate a 3D environment the way VR controllers or the Spacemouse can do.

2.5 Related Work

When experimenting with 'active' and 'passive' learning of an 3D virtual object's structure, *James et al. (2002)* defined the term 'wobbles' as 'movements with a range of $\pm 22.5^\circ$ around the central point for no more than 10 seconds' and that users tend to perform these wobbles with 3D objects around plan views to gain maximum

amount of information about the object's structure. *Viet et al. (2009)* studied the influence of degrees of freedom on object orientation in VR environments and concluded that simultaneous manipulation of all the DOFs does not necessarily lead to the best performances.

Using two kinds of interaction methods using two different hands can be called a hybrid mode of interaction. While experimenting with hand tracking and bimanual controllers for virtual reality, *Huang et al. (2020)* proves through user testing that hybrid interactions can reach the performance levels of using just two bimanual controllers, which is the default standard. *D Way et al. (2014)* conducted an interesting usability evaluation of free-hand microgestures using a wrist worn time of flight sensor. *Chan et al. (2016)* conducted user elicitation studies on single-handed microgestures and discussed its implications for design.

2.6 Problem Identification

The problems that were identified were as follows -

1. Natural object exploration in 3D requires at least 3 (ideally 6) degrees of freedom (DOF), a mouse only has two. The current method is to press and hold an

activator key like Shift/Alt/Ctrl and rotate along two axes or pan along two directions. This also forces the user to sometimes zigzag along axes and switch between pan and zoom multiple times to reach a certain camera view.

2. Dragging actions in CAD such as extruding often require very fine adjustment and users end up straining their hands clenching the mouse in order to make these movements. People would also like to see transformative actions like Extrude from multiple angles while they are doing them.
3. Users suggested they like using standard views but would like more angles to snap to. Some users also suggested they would like the ability to easily snap to custom views.
4. The navigation controls (activator buttons such as Shift/Alt/Ctrl and middle mouse button) have no semantic relationship with the actions that they do on screen.
5. When zoomed in, the mouse sensitivity is very low and it requires dragging the mouse multiple times to achieve the desired effect. There is also confusion about the pivot point which is at the centre of the object some times and at the origin at other times.

Ideation and Conception

3.1 Ideation

Based on the user study and secondary research, I came up with a few ideas that could address some of the issues and shortcomings or enhance navigation in CAD environments.

3.1.1 Dual mouse

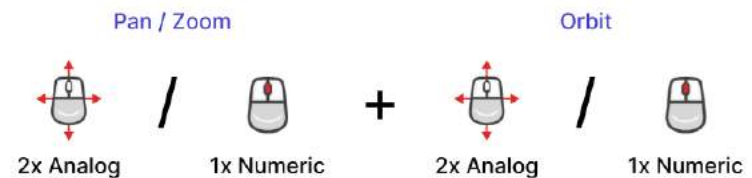


Figure 3.1.1 : Using two mice to have 3+ DOF

Since one mouse doesn't have the capability to deliver a full 3DOF navigation experience and requires the usage of activator keys (*Figure 2.4.1*), the idea is to use two mice, one on either side of the keyboard and have a control

scheme (Figure 3.1.1) that may simulate holding two VR controllers.

3.1.2 Foot based controller

Doldrum

Doldrums are a set of bowl like devices that go under the user's feet. They allow the user to perform actions by natural leg movements which are assisted by the self balancing design of the concept.

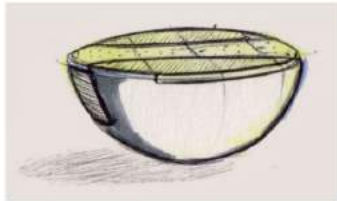


Figure 10: Doldrum

The user rests his feet on a memory foam top. The comfortable surface allows the feet to be heated for comfort in cold conditions as the device

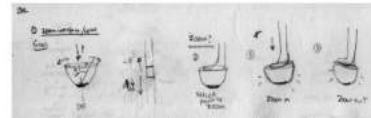
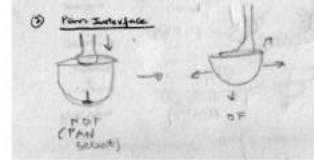


Figure 11: Doldrum - Zoom interface (top) and Pan interface (bottom)



Some other interactions menu activation, selection and manipulation may be possible using the same device.

Figure 3.1.2 : Doldrums concept by Gupta, R. (2018)

The ability to navigate the camera around the object in the middle of transformative actions can prove very useful to understand precisely what changes are being made, as listed in one of the identified problems. Rohit Gupta from IDC School of Design had an interesting concept in his P2 Project report which mentioned using doldrums (a half inverted sphere that feet can balance upon). The idea is to shift the balance of the doldrums using one's legs to control the camera accordingly, thus offloading the task of navigation to the idle limbs.

3.1.3 Grip sensitive DPI

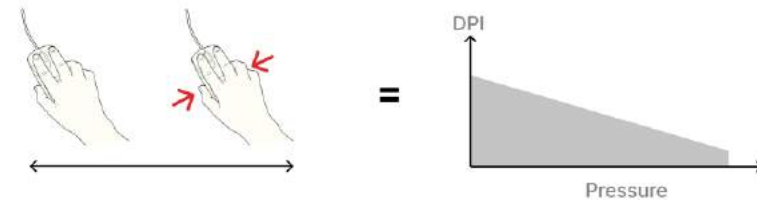


Figure 3.1.3 : Grip sensitive dynamic DPI

DPI or Dots Per Inch is a metric that dictates basically how sensitive the mouse is. As listed in the Problem Identification section, I noticed users tended to make minute adjustments in CAD using their mice, and would naturally clench their mice harder to do so. The idea here is to include force sensors where the mouse and little finger rest so that clenching harder will automatically drop the DPI of the mouse and make it easier to make minute adjustments.

3.1.4 Quick numeric entry



Figure 3.1.4 : Dimension Input Text Field in Fusion 360

I had observed that the most frequently input format of data in CAD software is in the form of `number.number unit` as shown in *Figure 3.1.4*. This was because most object manipulation actions are done by inputting dimensions in that format. Moreover, most of the time, users tend to use the default units, so that leaves the most frequent data entry in the format `number.number`. That gave me the idea of developing a way to quickly input numeric data.

3.2 Idea Evaluation

These ideas were evaluated through a technique called the Abstraction Ladder. Each idea is put through the test “What is the actual purpose of this?” or “What is the underlying problem that this is addressing?” and “Can I address a broader issue than this?” and map more abstract ideas higher on the ladder. This method can give rise to more holistic solutions or problem statements. For example, I started with the term ‘Navigation’ and asked “What is the purpose of this?” which led me up the ladder with the term ‘Object Exploration’, and further up with the term ‘Structural Understanding’, until I got to the concept of ‘CAD’ itself. Then I did the same to my solution ideas and mapped them to the terms on the main ladder as shown in *Figure 3.2.1*.

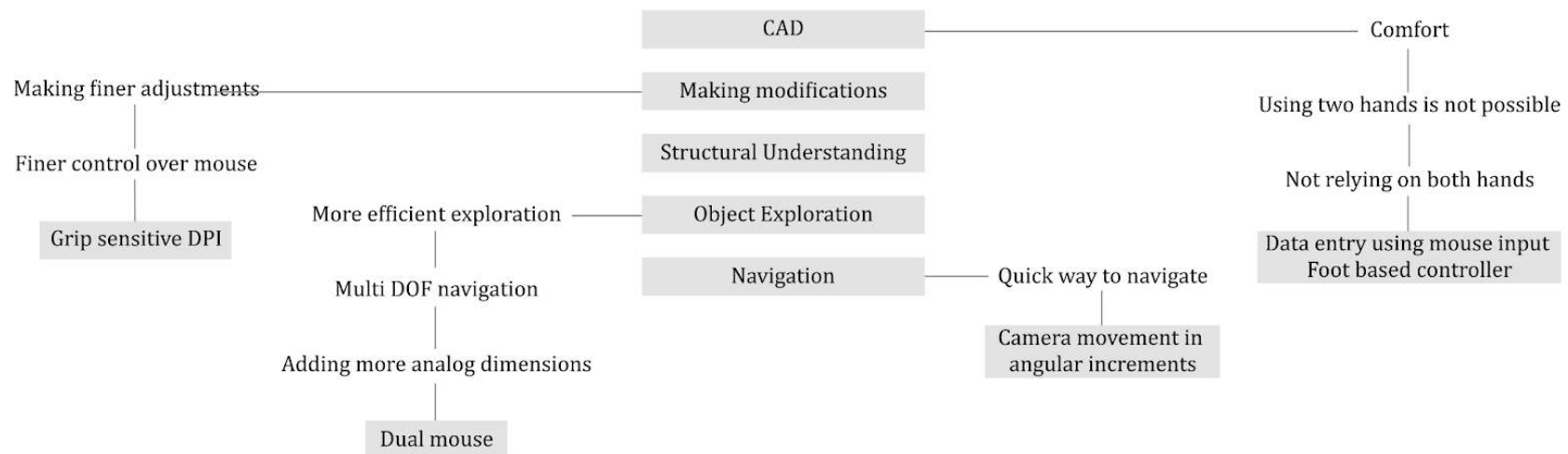


Figure 3.2.1 : Idea Evaluation using the Abstraction Ladder Technique

3.3 Restated Design Brief

After the evaluation, I concluded that instead of adding more degrees of freedom or adding niche navigation methods and making navigation more complex, I should focus on creating simple, natural and intuitive interactions. It also led us to a new area of focus - single-handed interactions. Whether a person only has one dexterous hand or is simply eating chips with one hand while operating the computer with the other hand,

single-handed interactions seemed like an interesting area with real world applications.

The computer mouse has stood the test of time because it is extremely precise and can be quite low cost. Therefore, we chose to narrow our focus to invent **interactions that supplement the human-mouse interaction** so that the mouse can be not just a pointing device, but rather, an intelligent, interactive object.

Note: The real world applications of the concepts can be generalized, however this report will only mention CAD applications as that has been the area of initial study.

3.4 Concepts

3.4.1 Pan/Rotate Microgestures

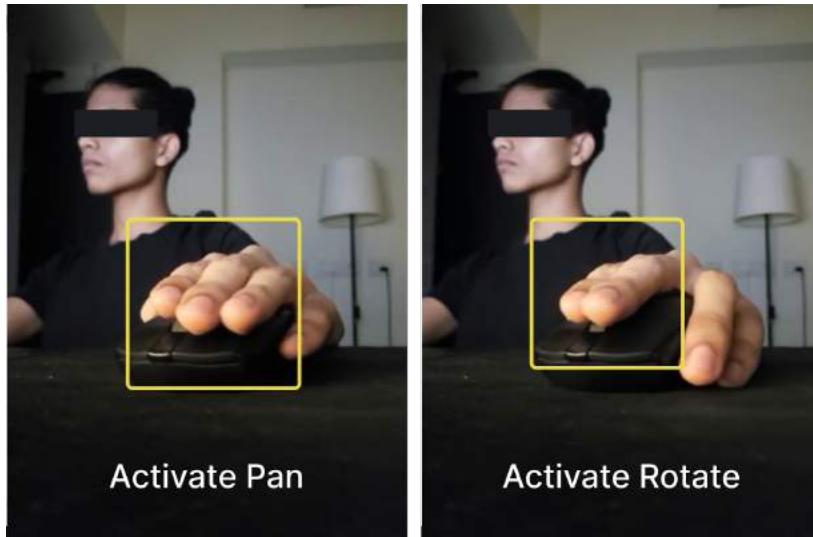


Figure 3.4.1.1 : Pan and Rotate Microgestures

Since Object Navigation happens using the mouse only after the user presses and holds activator keys on a keyboard, the single-handed interaction will naturally have to happen through the mouse alone.

Mapping basic commands and shortcuts to the mouse can prove very useful to users (*Trewin et al, 2009*), so we tried to come up with an interaction that doesn't interrupt the

mouse-holding posture of the hand much. Adding extra buttons was an option, as well as any other mode of interaction, but I chose to go with gestures because they can be implemented through either a novel mouse with dedicated pan/rotate buttons or using a simple webcam with an AI (Artificial Intelligence) based software that recognises the user's gesture. This way the user can use any mouse of their choice and wouldn't have to spend money on getting a new mouse (the Covid-19 pandemic has led to most people owning webcams anyway).

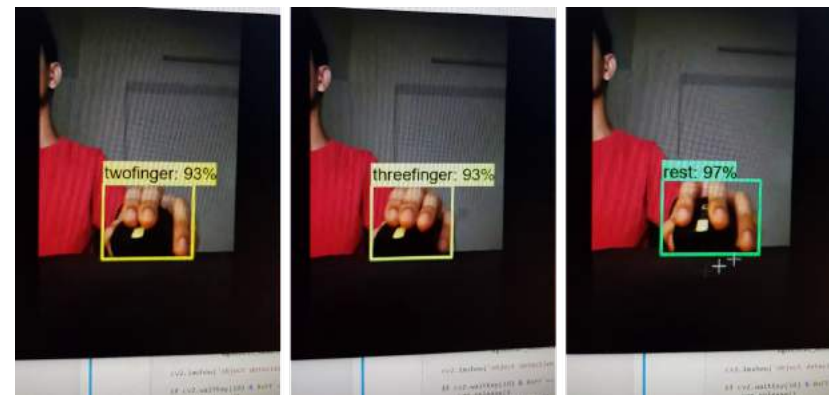


Figure 3.4.1.2 : Machine Learning based proof of concept

Figure 3.4.1.2 shows the proof of concept I programmed from scratch using *Google's Tensorflow APIs* and *Nicholas Renotte's Youtube course* on machine learning. I taught the AI to recognise the number of fingers lifted or the number of fingers connected using only a simple webcam. This

program can then be used to simulate the pressing of appropriate activator keys for CAD. The gesture two fingers connected can be mapped to Rotate as it is the most used function, and three fingers connected can be mapped to Pan as it feels like one has kept the hand on the mouse and is moving it to move the object on-screen; making the control mapping feel natural.

3.4.2 Gyroscope Mouse

Object Navigation in VR is drastically better with two controllers because one actually rotates the 6DOF controller to rotate the object and moves the controller to pan/translate the object. I wanted to bring such an interaction to the desktop CAD experience and make it require only a single-hand.

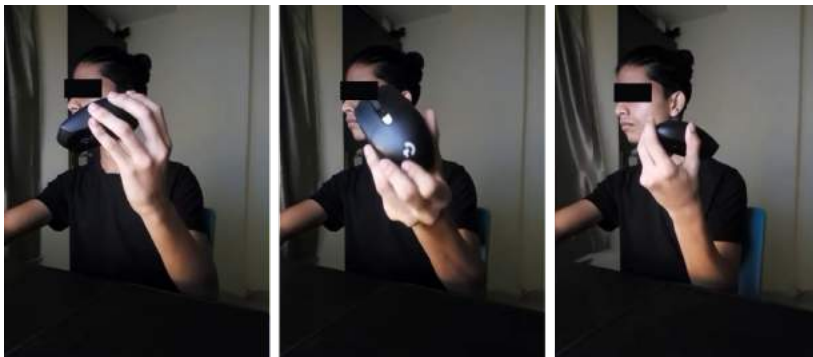


Figure 3.4.1.1 : Gyroscope mouse concept

The mouse detects when the user lifts it off the surface and positions it in a preset orientation (like the scroll wheel facing the user) and activates a 'Navigation Mode' where the orientation of the on-screen object is directly mapped to how the user orients the mouse. If this interaction is comfortable, then the user can also move the mouse closer or farther away from them and the object can zoom on-screen. Navigation Mode can end when the mouse detects that it has been placed back on a flat surface. It only makes sense to have this on a wireless mouse though.

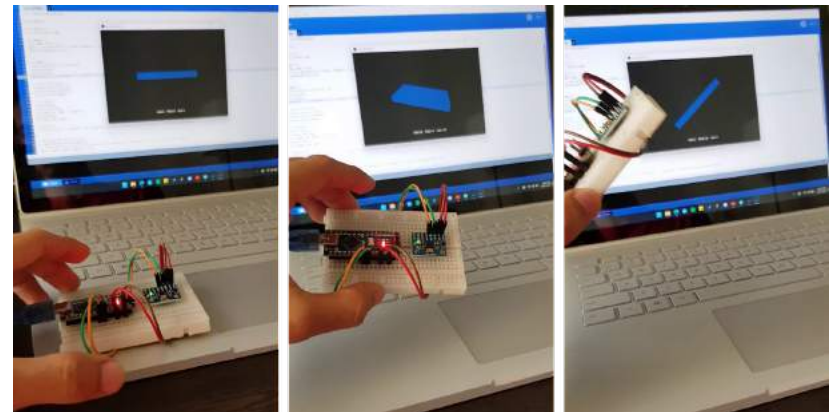


Figure 3.4.2.2 : Gyroscope navigation proof of concept

This interaction requires only embedding a low cost gyroscope on the hardware level and a driver software on the software level. Figure 3.4.2.2 shows the proof of

concept I made using a low cost MPU6050 accelerometer/gyroscope and an Arduino Nano.

3.4.3 Fingerspelling Numeric Input

I noticed most of the data entry in CAD is for entering values, which are numeric. Apart from using activator keys, that is mostly what an average user (one who doesn't use hotkeys) uses the keyboard for. An average user also wouldn't want to remap custom shortcuts to buttons which have no semantic correlation to those numbers.

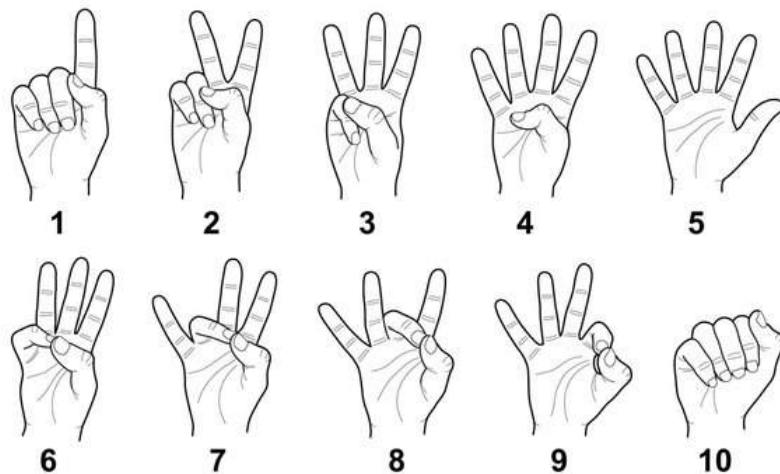


Figure 4.1.1 : Numbers in Sign Language

Thus it made sense for me to come up with a control scheme that either was already widely accepted or one

that didn't require much memorisation or guesswork. I studied the number gestures in Sign Language (*Figure 4.1.1*) as I thought it would be better to stick to the universal standard. The problem with these, beyond 5, was that it required touching the thumb to the other fingers one at a time. This meant the **standard sign language number gestures couldn't be used as hybrid gestures**.

After some brainstorming, I came up with using Fingerspelling as a means for data input. Fingerspelling upto the number 5 should be fairly intuitive. And if there is a clear governing logic then the learning of gestures above number 5 should also be easy.

I used the same ML program from the Pan/Rotate microgestures concept to test out the feasibility of fingerspelling gestures on top of a mouse as hybrid microgestures (alongside the primary interaction of an input device).

3.5 Chosen Concept

I evaluated the three concepts based on feedback from my jury, as well as scope and novelty. While the first two concepts seemed like gimmicks, I felt that Fingerspelling

Numeric Input was novel and had broader applications than just CAD. It was a data input technique in and of itself.

While we had explored these other concepts, **we decided to broaden focus from CAD only and develop and evaluate fingerspelling as a text input technique.**

Design and Detailing

4.1 Description

Fingerspelling gestures are a set of interactions that allow a user to input numbers from 0 to 9 and a decimal point (.) through various combinations of finger taps using a single hand. These gestures are virtually independent of finger position in space, as they are governed only by a) whether a particular finger was lifted or not and b) which fingers were lifted. This allows the gestures to be executed while holding physical objects such as computer mice without interfering with their function. Thus, a user can have a hybrid mode of input where a single hand can use the mouse as a pointing device as well as a numeric input device.

4.2 Gestures

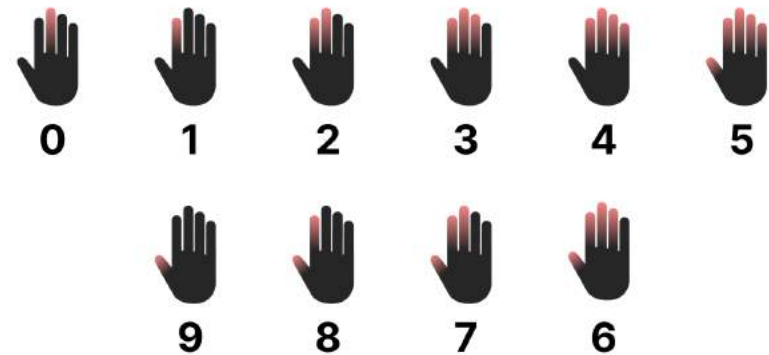


Figure 4.2 : Fingerspelling gestures encoding scheme

Since we have 5 digits on our hands, it is straightforward and intuitive to spell numbers till 5 -

1 : lift index finger

2 : lift index and middle fingers

3 : lift index, middle and ring fingers

4 : lift index, middle, ring and little fingers

5 : lift thumb, index, middle, ring and little fingers

For numbers greater than 5, the logic is to reverse the series of lifting fingers -

6 : lift thumb, index, middle and ring fingers

7 : lift thumb, index and middle fingers

8 : lift thumb and index finger

9 : lift thumb

It can also be thought of as lifting the opposite number of fingers, but instead of keeping the thumb down, keep the little finger down.

Since zero is very frequently used (which is why the zero keys on calculators are often bigger), it had to be mapped to an easy gesture even though it didn't have semantic correspondence. Same was the case with the decimal point.

0 : lift middle finger

. : lift middle and ring finger

4.3 Working Mechanism

As these gestures are designed to be hybrid, it has to be clearly understood what counts as a gesture and what does not. To prevent accidental keypresses through these gestures, I came up with an algorithm that the system uses as a safety net or filter to make sure the user intends to do the gesture -

1. Check if all five fingers are planted and set all finger counters are set to `false`
2. If a finger is lifted, set its counter to `true`
3. Wait for all fingers to be planted
4. If all 5 fingers were planted -
 - a. If it has been more than the `timeout` time-
 - i. Set all counters to `false`
 - ii. Return to Step 1
 - b. Analyse the true-false permutation of finger counters, and register the corresponding keypress
 - c. Set all counters to `false`
5. Return to Step 1

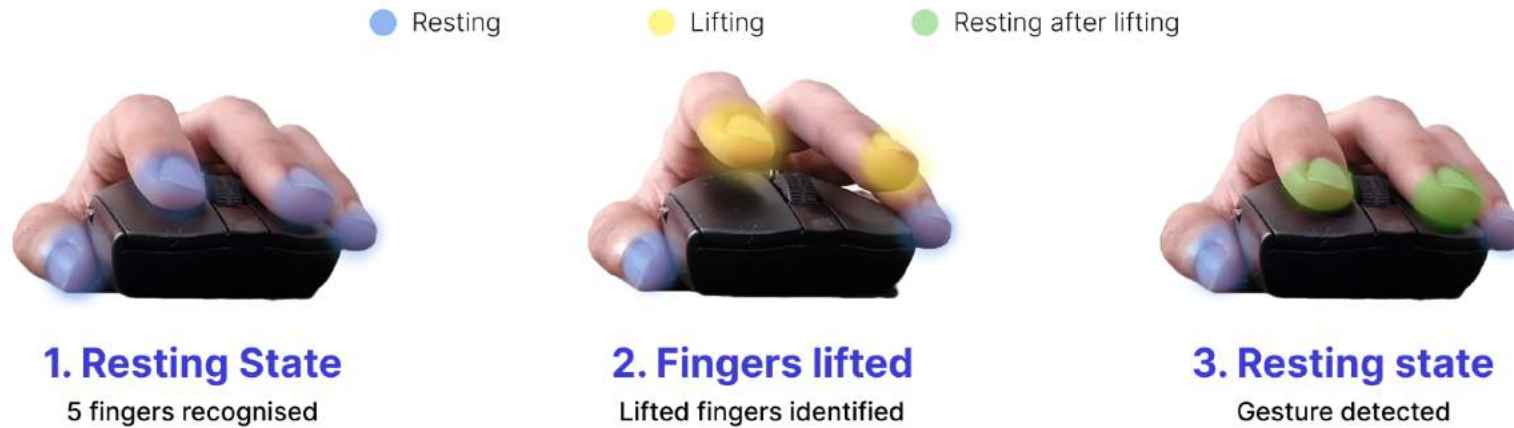


Figure 4.3.1 : How the Fingerspelling Hybrid Gestures work

The `timeout` time had to be enforced to make sure it doesn't register a keypress if the user lifts their hand up and places it back. This protocol requires the user to have their hand in the resting state (all five fingers planted) before doing any gesture. So if the user is holding a mouse, all of their fingers need to be planted before they can activate any gestures.

This protocol opens up possibilities for any multitouch device to implement these gestures as it can calculate gestures like so -

1. If there are 5 points of contact (5 fingers pressed) -
 - a. Expect fingerspelling gestures

- b. Register all 5 points of contact
2. If some points of contact disappear (some fingers were lifted), note which fingers were lifted
3. If all points of contact disappear before t time, proceed to the next step, else return to Step 1
4. Analyse the true-false permutations directory and register a keypress
5. Return to Step 1

Evaluation

5.1 Apparatus

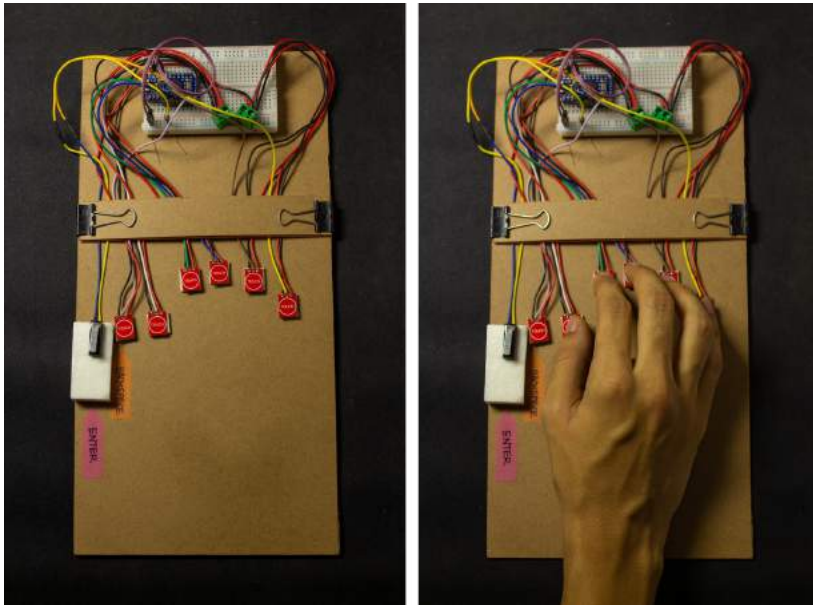


Figure 5.1.1 : Fingerspelling Touchpad Rig

Two working prototypes were built - one to simulate a touchpad (*Figure 5.1.1*) and one to emulate a mouse (*Figure 5.1.2*). The touchpad and mouse rigs use six TTP223 touch sensors and a momentary tactile connected to an Arduino Pro Micro that uses the Keyboard.h library

to emulate a HID (Human Interface Device) on Windows 11 running on a Microsoft Surface Book 2 laptop. The typing test programs were coded from scratch in Python using the Tkinter interface. For Test 2, the keyboard used was a Logitech K230 and the mouse used was a Logitech G304.

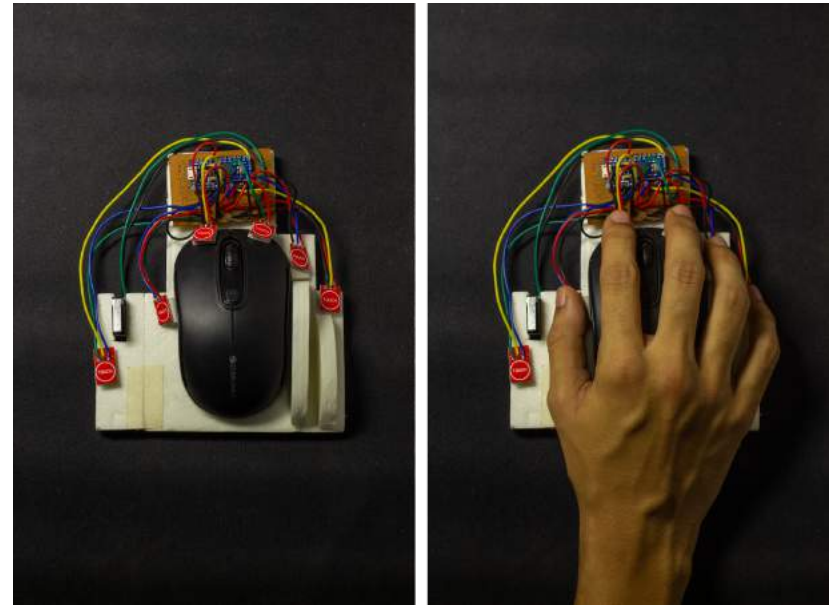


Figure 5.1.2 : Fingerspelling Touchpad Rig

5.2 Limitations

The TTP223 capacitive touch sensors had a major issue;

When multiple sensors are connected to the Arduino and multiple fingers are resting on them at the same time (the default state of the interaction is all fingers resting on all sensors) for 4-8 seconds then they start randomly activating. The only fix for this was to lift all fingers which instantly solves the issue, but causes hindrance in the typing test.

These sensors also activate even if a finger is sufficiently close to them, which means there was a lack of pinpoint accuracy that would have been ideal for a study of microgestures. The coding overall could have been better calibrated that could have led to different results.

The size of the touch sensors was also too small, coupled with a standard one size fits all prototype rig as well as a standard office chair and desk which must've meant a good portion people didn't have ideal ergonomics, which might have affected their performance. **The ↵ (Enter) and ← (Backspace) keys in these rigs are just a means to an end**, the physical implementation will depend upon the ergonomics of that particular device.

Due to time constraints, I could only test the fingerspelling rigs and not do consecutive tests with conventional numeric input devices such as Numpads and Numrows, so while the test does one test of the Numrow and Numpad

per person in the end, this test does not venture the comparison of the learning curve of my rigs with conventional numeric input devices. Finally, this typing test has its inherent limitations as it doesn't evaluate real world use but is rather a synthetic benchmark.

5.3 Methodology

The evaluation employs an experimental design and evaluation methodology for gathering statistical usage data and qualitative data collection through semi-structured interviews and observation.

5.4 Demographics

All participants were students at IDC School of Design, IIT Bombay with ages ranging from 20 to 23. There were 11 male participants and 4 female participants. All of the participants were right handed and fully dextrous in both their hands.

5.5 Metrics

The time difference between starting a session and ending a session were recorded in seconds. Each phrase was marked as correct or incorrect based on the data recorded after pressing Enter, and the correct percentage was calculated. There were 30 phrases in the session with a total of 57 characters. Therefore the total number of characters required to to complete the test was 57 + 30 Enter keys = 87.

The Characters Per Minute (CPM) for each test was calculated using the following formula:

$$CPM = \frac{(Percent\ correct)*(Total\ characters)*60}{(Time\ in\ seconds)}$$

The Error Rate (ER) was calculated by calculating the percentage of the ratio of Backspaces pressed to the total number of characters:

$$ER = \frac{(Number\ of\ Backspaces\ pressed)*100}{(Total\ characters)}$$

5.6 Test 1

5.6.1 Goal

The goal of Test 1 is to evaluate the learning curve of the Fingerspelling gestures as a standard input method in terms of metrics like CPM (Characters per Minute) and ER (Error Rate).

5.6.2 Procedure

The gestures were demonstrated to each participant. They were made to imitate the researcher's hand for each gesture and the system level working of the gestures was explained for better clarity. They were then allowed to practice the gestures on a blank text document until they were comfortable.

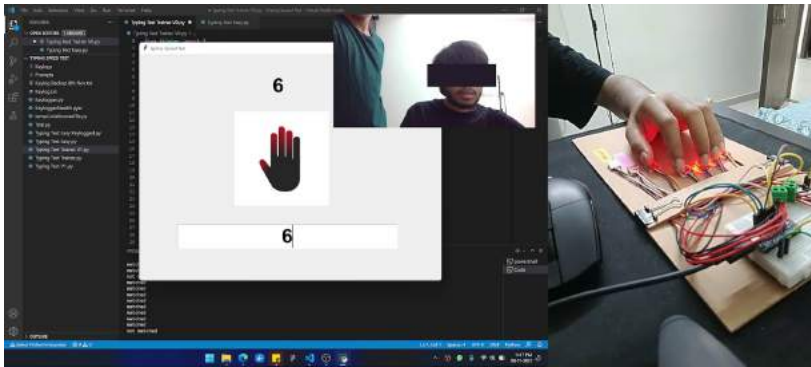


Figure 5.6.2.1 : The Trainer Program

This was followed by a trainer program, which displayed a phrase along with an illustrated graphic of what fingers the participant is supposed to lift, and a text field below to enter the phrase. They were allowed to use the trainer program until the participant said they were comfortable to do the test without the graphic. All this was done only for the first session.

Each such session was recorded along with two camera feeds - one of the participant's face (to know whether they were looking at the screen or the rig) and one of the hand placed on the rig. The hand recording was mirrored and placed beside the screen capture in sync to make it easy to interpret the recording. All keystrokes in the test along with the time of keypress were logged in the host computer.

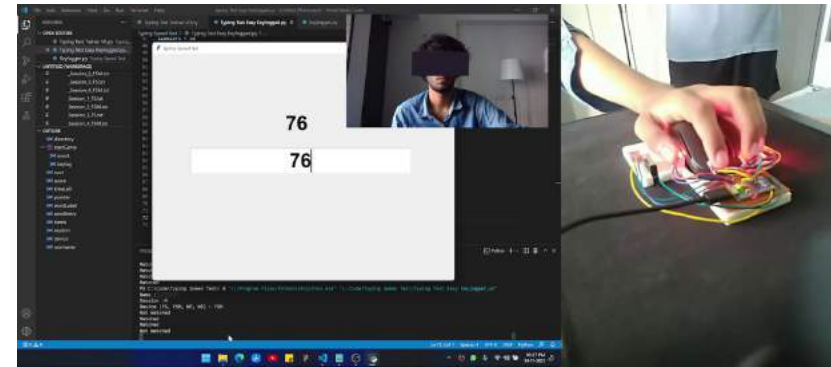


Figure 5.6.2.2 : The Typing Test Program

Finally, the typing test program was opened and users were asked to input the number being displayed on screen into a text field, and press the ↵ (Enter) key on the rig to go to the next number. The typing test was the same throughout the evaluation period and had a total of 30 preset phrases per session which were displayed in random order each time. The corpus (directory of phrases used during the test) was as follows:

```
[ '23', '41', '63', '82', '37', '39', '76',  
'40', '24', '1', '2', '3', '4', '5', '6',  
'7', '8', '9', '10', '0', '54', '84',  
'82', '49', '58', '93', '522', '744',  
'123', '323', '100' ]
```

Participants were asked about their preference between the touchpad rig and the mouse rig, and the sessions were

alternated between the two. The first few sessions of each participant were followed up with semi-structured interviews.

5.7 Test 2

5.7.1 Goal

The goal of Test 2 is to compare the fingerspelling technique as an open and closed loop input method with standard input methods.

5.7.2 Hypothesis

An open loop system may be defined as a system where 'the state of the system is affected by the input of a desired quantity, but the system does not adjust itself dynamically to keep the state close to the desired state' (Cook P. R., Princeton Real-time Lecture Notes, 2004) ie. the user can't see what key they are about to press. Conversely, a closed loop system is one where the user dynamically adjusts the

placement of their finger mid air to accurately land on the desired key.

One of the main reasons the fingerspelling concept was chosen as hybrid gestures was because while using CAD on desktops using a single hand, the user has to switch the hand between mouse and keyboard very frequently. Even though key positions can be memorised on the keyboard, the relative position of the mouse and keyboard is seldom the same. That would mean the user has to look away from the screen down at the keyboard everytime they switch.

Test 2 was designed with the single-handed data input + mouse movement interaction in mind. Given the fact that the Fingerspelling Mouse Rig doesn't require the hand to move over to the keyboard, it was speculated that the Fingerspelling Mouse Rig would do well in Test 2.

5.7.3 Procedure

The test was done after participants had done 6 or more sessions of Test 1 and had memorised the gestures.

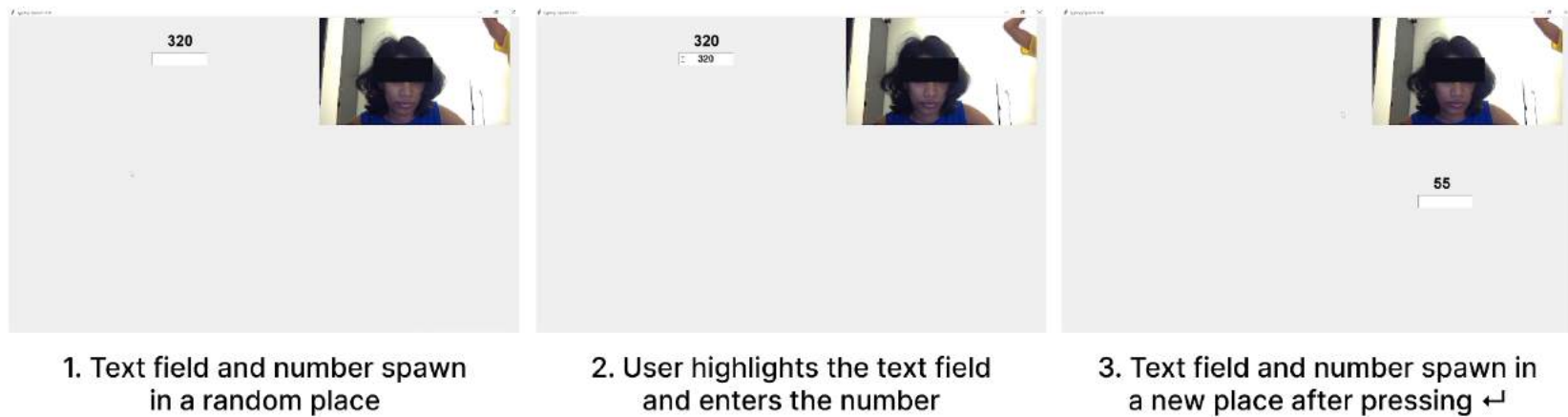


Figure 5.7.3.1 : Working of the Test 2 program

Participants were asked to use three different modes of input - a Numpad, a Numrow and the Fingerspelling Mouse Rig. They were shown a number with a text field below it as seen in *Figure 5.7.3.1* and were required to highlight the text field by hovering the mouse pointer over it and clicking the left mouse button (The text field did not accept input unless it was highlighted using a mouse click. This mechanism was enforced to simulate the frequent mouse-keyboard switching in single-handed CAD use), and then enter the numeric phrase on-screen using the given mode of input. To submit a phrase, the participants were required to press enter, after which the number and text field would move to a random new location on-screen.

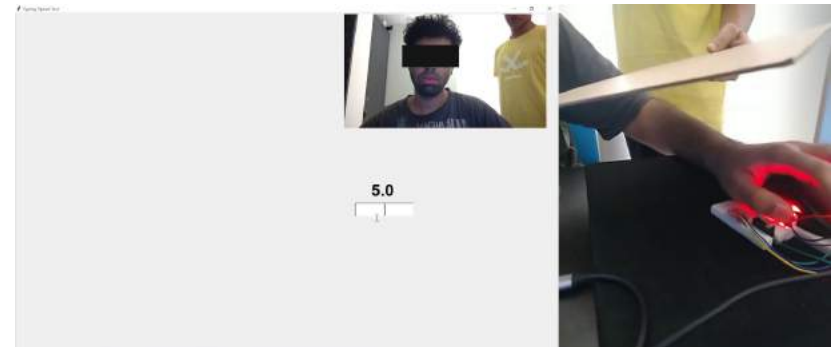


Figure 5.7.3.2 : The keys/rig are covered for the Blind test

This was repeated with 20 phrases, and counted as 1 test. 6 tests were done per participant - Numpad Sighted, Numrow Sighted, Mouse Rig Sighted, Numpad Blind, Numrow Blind and Mouse Rig Blind - in that order. For the

Sighted tests, participants were allowed to look at the keyboard and mouse while using it, as they usually would. For the Numpad and Numrow Blind tests, the keyboard was covered from the participant's sight using a sheet of MDF board, however they were allowed to look at the mouse. For the Mouse Rig blind tests, the rig was concealed using the same sheet. Completion times were recorded. The following corpus was used:

```
[ '10', '20', '30', '40', '50', '55', '65',  
'75', '100', '2000', '10', '50.5', '5.5',  
'7.6', '9.2', '6.5', '4.3', '500', '320',  
'5.0', '2.0' ]
```

A lot of instances of 5, 0 and . (decimal) were used to simulate an actual CAD environment as dimensions are usually in multiples of 10 and 5 with frequent use of decimals.

5.7 Difficulty Rating Survey

To evaluate the difficulty of each gesture, a 5-point Likert scale was administered. Users were asked to rate each gesture on a scale of 1 (easiest) to 5 (hardest). The ratings were meant to be relative, meaning that at least one gesture will have a 1 score and at least one gesture will

have a 5 score. The arithmetic mean and mode were calculated and plotted using Microsoft Excel.

Results

6.1 Test 1 Results

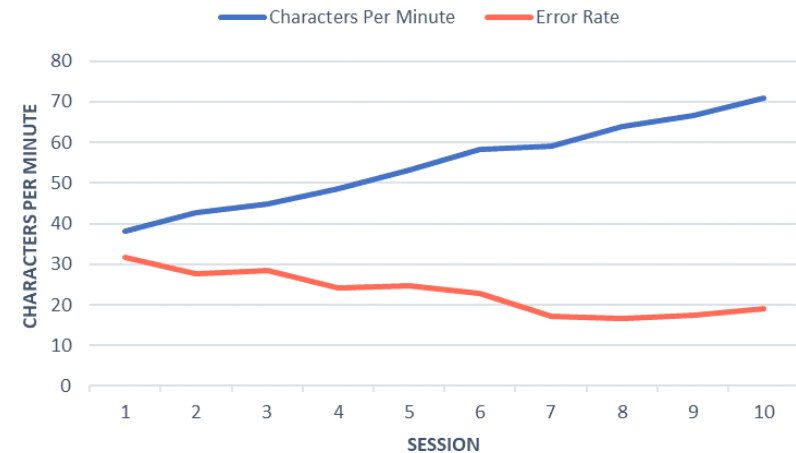


Figure 6.1 : Mean CPM and ER scores of 10 participants

10 participants each completed 10 sessions, making a total of 100 sessions of data for Test 1 results. The mean CPM for Session 1 was 38.12 and the mean CPM for Session 10 was 70.96. The mean ER (Error Rate) at the beginning was

31.7% and it was 19.1% at the end, the lowest being 16.5% in the eighth session.

Disclaimer : The rigs would oftentimes accidentally register a Backspace keypress right when the participant was about to press Enter. This would lead to most phrases being recorded as incorrect even though it's not human error. Such cases were noted and those occurrences were counted as correct.

6.2 Test 2 Results

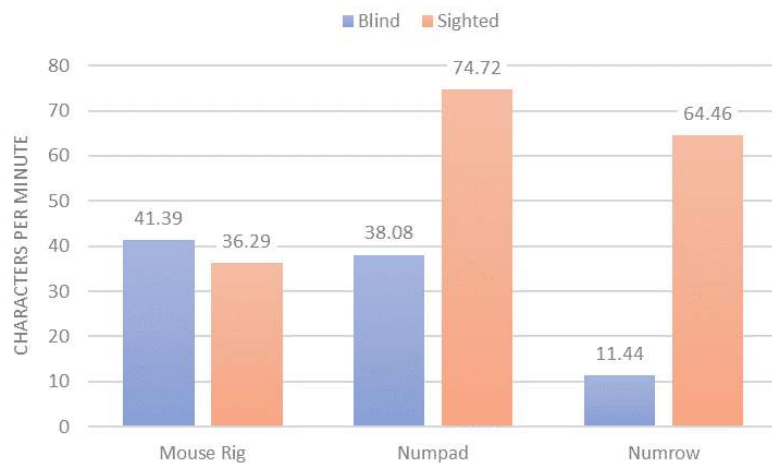


Figure 6.2.1 : CPM comparison for Test 2

10 participants completed 6 tests each for Test 2. The mean CPM (Characters per Minute) score comparison for Sighted and Blind tests conducted during Test 2 are shown in *Figure 6.2.1* and the ER (Error Rate) comparison for the same is shown in *Figure 6.2.2*.

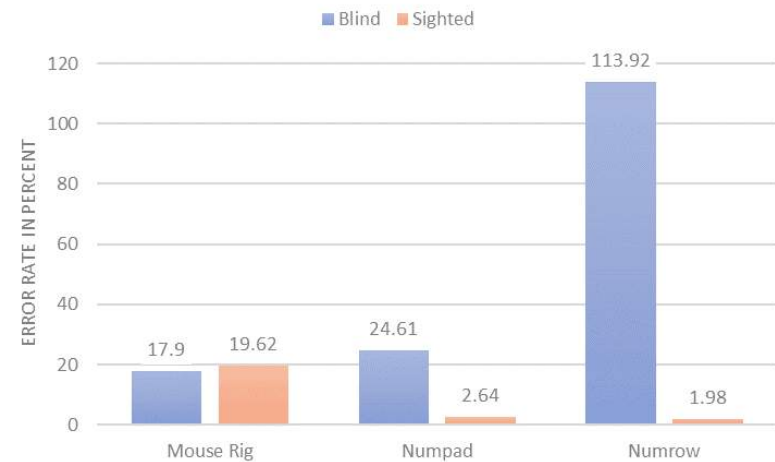


Figure 6.2.2 : Error Rate comparison for Test 2

The Mouse Rig scored 41.39 CPM on the Blind test as compared to 38.08 CPM of the Numpad and 11.44 CPM of the Numrow while it scored 36.29 CPM on the Sighted test compared to 74.72 CPM of the Numpad and 64.46 of the Numrow.

The ER for the Mouse Rig is 17.9% for Blind and 19.6% for Sighted tests while the ERs for the Numpad are 26.61 and

2.64 respectively. Finally, the ERs for the Numrow are 113.92 and 1.98 respectively.

Disclaimer : The Mouse Rig would oftentimes accidentally register a Backspace keypress right when the participant was about to press Enter. This would lead to most phrases being recorded as incorrect even though it's not human error. Such cases were noted and those occurrences were counted as correct.

6.3 Difficulty Rating

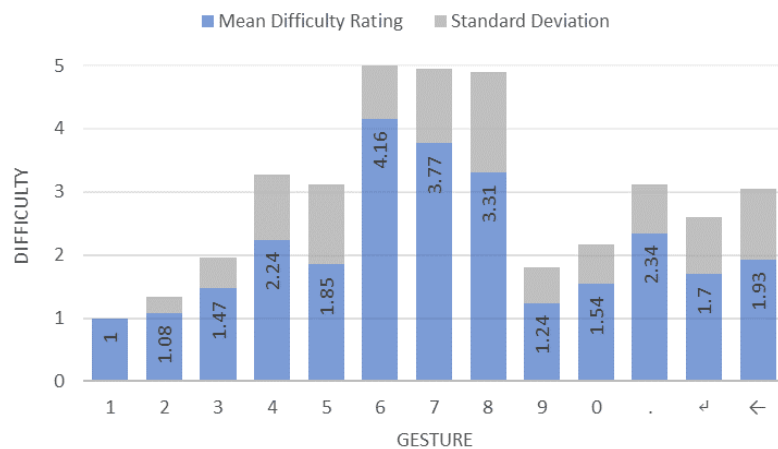


Figure 6.3.1 : Mean and Standard Deviation of ratings

Mean scores and Standard Deviation (SD) of the difficulty rating Likert Scale questionnaire are shown in *Figure 6.3.1*. The sample set for these studies was 16 participants. The gestures for 1, 2, 0 and 9 are the ones which require only one finger to be lifted (apart from 2), and are also rated as the easiest. Similarly, the gestures for 6, 7 and 8 have two levels, thinking what their opposite number is and lifting multiple corresponding fingers and were rated the hardest. Even though 5 requires all fingers to be lifted, it was rated easier than 4. Even though 0 was not intuitive it was rated as easier because it only required lifting one finger.

6.4 Interview Findings

6.4.1 Gestures

Participants memorised the gestures fairly quickly. Some participants could even type their phone numbers into a blank text field before using the trainer application. There was virtually minimal or no difference between touchpad and mouse rigs even in the initial sessions. However, some participants initially had trouble lifting fingers 'that accurately'.

Count	Feedback
10	Memorisation of gestures is fairly easy
5	I can get better at this over time
3	I knew the gestures but my fingers need better muscle memory to execute them

6.4.2 Encoding Scheme

Participants had trouble memorising 6, 7, 8 and 9. They said they required a lot of practice to execute and even then, it was cognitively taxing. Over 5 participants said it was more intuitive to invert the encoding above 5 (lift just one finger for 6 rather than four, two fingers for 7 rather than three and so on). Participants also made errors when consecutively executing gestures that required the same number of fingers to be lifted (like 64, 73 and 82). One participant said the 0 gesture (lift middle finger only) was counterintuitive since 0 is 'at the end of the number line' but it made sense since entering 0 is required frequently. Another participant said they would prefer it if they had to lift only one finger per gesture. Only one participant said the encoding scheme above 5 was intuitive.

Count	Feedback
7	I have trouble doing 6 / 7 / 8
4	Numbers above 5 are confusing and counterintuitive
4	Numbers below 5 are simple

6.4.3 Ergonomics

Over three participants said it was cognitively harder for them to lift their fingers for a gesture rather than press them. Three participants said it was difficult for them to locate the sensors after they lifted all the fingers without looking. Few participants initially lifted all fingers after each gesture, and few continued to lift all fingers for pressing Enter or Backspace.

One participant said the mouse rig was too small for their hands while another said the rig was too big. Three participants said having to bend their fingers to hold the mouse rig was uncomfortable and affected their performance. Over 4 participants said even though the touchpad rig has a lack of palm support, their fingers feel more relaxed and they can see where they are placing their fingers; most participants had to look over and

around the mouse to find the sensor for the little finger when placing their hand for the first time. Three participants would often click the mouse button when doing gestures involving the index and middle fingers.

Count	Feedback
6	I can't know if my little finger is actually planted on the mouse rig
3	The mouse button clicking when doing gestures is annoying
3	It is cognitively hard for me to lift fingers for a gesture rather than pressing them down
3	I can't locate the sensors without looking after I've lifted all my fingers

6.4.3 Mouse v/s Touchpad

Some participants strongly preferred the mouse rig over the touchpad rig because it offered better support to their palm and fingers. Other participants felt the touchpad rig was more comfortable because they could clearly tell the position of their fingers and that it was more relaxed. Overall, the performance on both the rigs was almost

identical but some people preferred mouse ergonomics while others preferred touchpad ergonomics.

A lot of participants confused Enter and Backspace keys often. Most participants said Enter and Backspace were too close, but agreed it was necessary to have them both near the thumb. One participant said they would like to have them mapped to the mouse's side buttons. Most participants misclicked Enter or Backspace, or accidentally activated other gestures while trying to press either. Some participants would lift their fingers completely while pressing either, and that resulted in fewer misclicks at the cost of typing speed. However, over 4 participants said they could tell Enter and Backspace apart because one was a physical momentary switch whereas the other was a touch sensor just like the other finger sensors. Two participants said they liked the orientation of the Enter keys (Downward incline away from the hand) because they could pull their thumb back to press the switch.

Count	Feedback
8	Enter and Backspace are too close and are causing a lot of misclicks
4	Having different kind of buttons for Enter and Backspace definitely helps telling them apart

5	I prefer the touchpad rig over the mouse rig
4	I prefer the mouse rig over the touchpad rig
2	I like the incline direction of the Enter switch

Discussion

7.1.1 Learning Curve

It can be seen in *Figure 6.1* that the CPM and ER scores of participants steadily increased as they built up muscle memory of the gestures. It is almost a linear increase in typing speed, and it can be concluded that 10 sessions is far too few to predict where the learning curve will plateau. The reason for the plateauing and subsequent increase of the error rate approximately past session 7 is that once some participants got used to gestures, they would race the clock to finish the test, and would end up making more errors.

7.1.2 Gestures and Encoding

It is observed that after explaining the logic behind the encoding scheme of gestures, they are overall simple to

memorise, but it cannot be said that numbers above 5 are intuitive. As shown in the difficulty rating results (*Figure 6.3.1*) the distribution of gestures is uneven, and for higher numbers it directly correlates to the number of fingers users have to lift at once.

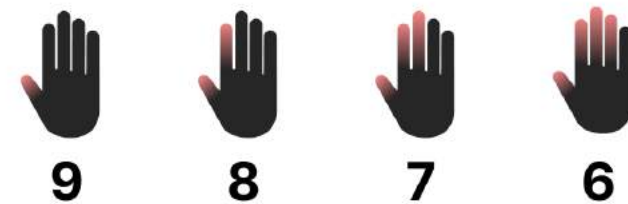


Figure 7.1.2.1 : Current encoding scheme above 5

The original encoding logic (*Figure 7.1.2.1*) was to find the opposite number in the decimal system to say, 6; 10 minus 6 is 4 so the user lifts 4 fingers, for 7, 10 - 7 is 3 so the user lifts 3 fingers and so on. The gestures for 6, 7 and 8 are the most cognitively and physically taxing since one first has to think about the reverse/opposite number and then lift multiple fingers to execute them.

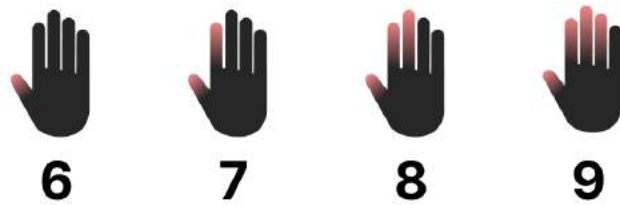


Figure 7.1.2.2 : Suggested encoding scheme above 5

Instead, as 4 participants suggested (*Figure 7.1.2.2*), it was more intuitive to reverse the encoding above 5 so that the higher the number, the more fingers one has to lift. This also makes sense using another argument that the gesture for 7, for example, is lifting the thumb and index finger, ie. lifting two fingers, which is more intuitive since it semantically means $5 + 2 = 7$.

Neither of these schemes, however, address one valid feedback by a participant which is that 0 should be at the 'end' of the gesture series since it is at the end of the number line. Also, one needs quick access to the zero gestures so it could be to lift the thumb only, but that doesn't sit well with any encoding scheme I could come up with. The rigs were also not without their bugs, and it would often be the cause for errors. The results could probably be significantly better if the rigs were better built and calibrated.

7.1.3 Ergonomics

Initially it was harder for participants to lift their fingers to do an action rather than press them down, but if these gestures are meant to be used on a conventional trackpad or mouse where the primary action aren't these gestures, then it is necessary to go through the learning process since pressing down will do the primary action of right and left clicks. These rigs shouldn't be used to judge the ergonomics of Fingerspelling gestures as that was out of the scope of this project and no ergonomics principles were actively applied other than 'this is what seems comfortable'.

This evaluation does reveal, however, that if Enter and Backspace are to be positioned near the thumb, then there needs to be some differentiation between the kind of switches/sensors. It is very confusing to remember and differentiate between Enter and Backspace since both are activated using the thumb. The initial prototype had both Enter and Backspace as touch sensors identical to the finger sensors, and there was a massive amount of misclicks between Enter and Backspace. The Enter key was promptly switched out with a clicky momentary switch and participants instantly appreciated the difference. This does not pose a major issue for the mouse implementation as a major fraction of mice available in the market already have

side buttons and they are well received. Another problem with the mouse rig is that participants would often click mouse buttons while doing the fingerspelling gestures. This can be resolved by either using firmer switches in a mouse or having a dedicated area for the index and ring fingers to rest on, and another area for right and left click.

Even the participants who preferred the mouse rig over the touchpad rig would find themselves looking over and around the mouse trying to find the sensor for the little finger and carefully placing it on the sensor. This can very easily be solved using bigger, prominent touch areas.

The touchpad rig was reviewed as more relaxed but because of the small sensor size and standard layout, participants with bigger hands found that they were cramming their fingers together. This shouldn't be a problem at all because as long as a touchpad is large enough to accommodate the user's fingers, they are free to place it anywhere they want.

7.1.4 Test 2

Test 2 simulates the kind of real world use that the Fingerspelling Mouse Rig was meant to be used in. While it is significantly slower than using a Numapd and Numrow single handedly when the participants move their eyes off

the screen and look what keys they are pressing, the Mouse Rig vastly outperforms the Numrow and even has an edge over the Numpad when it comes to CPM. The mouse rig also has the lowest ER in the blind test as compared to the other two. Another thing to note is that there are vast differences between the the Blind and Sighted test results of the Numpad and Numrow, whereas the results of the Mouse Rig are quite similar. Most participants even said they never look at the mouse rig anyway, but this was the first time they were using it as a pointing device and a fingerspelling rig simultaneously.

Conclusion

8.1 Conclusion

Instead of mapping custom gestures to each finger without semantic correlation and memorising them, we anticipated it was easier to learn an encoding scheme and have relatively harder-to-execute gestures. Evaluation does show it is easy to learn gestures based on a clear governing logic.

In real world applications, while switching between mouse and keyboard might be quicker for numeric entry, it requires taking one's eyes off the screen to locate the keys. Test 2 proves that this problem is solved by the Fingerspelling Mouse Rig since there is almost no difference between the Sighted and Blind tests.

Even on an imperfect rig, participants got used to getting around it's bugs and reducing their error rates and increasing typing speed. The ergonomics need to be optimised and the encoding scheme made intuitive. Especially for numbers below 5, where the gestures are intuitive with almost no learning curve, fingerspelling seems like a viable single-handed interaction method.

8.2 Additional Explorations

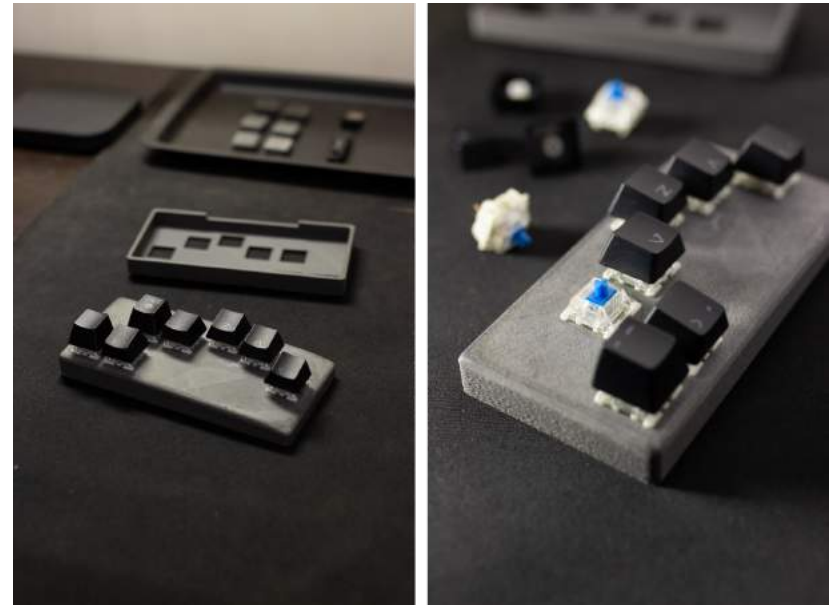


Figure 8.2.1 : Fingerspelling Keyboard Rig

After getting feedback from participants about how it is intuitive to press fingers down rather than lifting them up and how tactile feedback would be helpful, I 3D printed a third rig that uses Gateron Blue Clicky Switches that are usually used in mechanical keyboards. I made it because I was interested to see how a very tactile key would affect the performance and error rate, however, this was just one week before the report submission for my P1 Project and

doing a really brief evaluation wouldn't do justice to the technique, especially since the other two rigs have had over 120 sessions of evaluation. This can, however, prove useful when designing a mouse or other input device based on fingerspelling interactions where the focus is on tactile interactions.

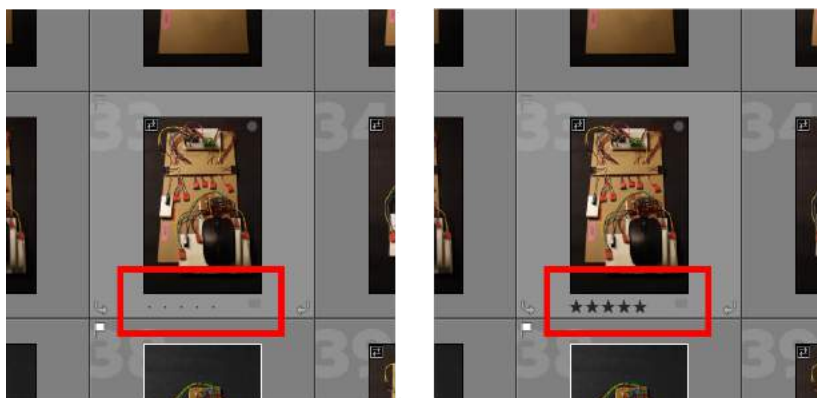


Figure 8.2.2 : Fingerspelling applied to UI elements

Fingerspelling can be applied to UI elements. For example, Windows has a shortcut for opening apps pinned to the taskbar - [Windows key + (n)] for the nth app on the taskbar; Fingerspelling can be used to trigger these shortcuts. It can also be used to select text styles in text editors such as Google Docs and Microsoft Word; a user can highlight text and the gesture for 1 can apply the style Heading 1 to a highlighted text and the gesture for 2 can apply Heading 2 and so on. It can also be used to rate

images on a scale of 1 to 5 while quickly scrolling through the image gallery in apps such as Lightroom. As shown in *Figure 8.2.2*, a user can rate an image by simply pressing the numbers 0 to 5, and with the Mouse Rig, there is no need to use another hand to press keys on a keyboard. I tried mapping these shortcuts to the rigs and they work as expected.


8.3 Future Scope

There are other physical implementations of these gestures - using a smartphone as a fingerspelling trackpad, a wearable device that recognises individual finger movements to discreetly take input using fingerspelling, a Leap Motion like hand recognition device specifically designed for fingerspelling, etc.

This technique particularly has scope in developing interactions for non visual interfaces such as a completely audio based interface for a music player, it can be used for shortcuts such as speed dial, for interacting with a device discreetly if implemented through a wrist wearable device or as a way to enter a security pin.

With devices like the Apple Watch already being able to recognise finger movements using the heart rate sensor and gyroscope, the possibility of consumer smartwatches recognising microgestures might not be too far fetched, given the technology already exists (SensIR, McIntosh et al, 2017). Fingerspelling Numeric Input **can prove useful as the primary interaction for actions like menu selection in smartwatches**, where on-screen lists don't exceed 5 items anyway.

References

1. Apple (Ed.). (2021, September 25). *Use assistive touch on your Apple Watch*. Apple Support. Retrieved November 24, 2021, from <https://support.apple.com/en-in/HT212760>.
2. Chan, E., Seyed, T., Stuerzlinger, W., Yang, X.-D., & Maurer, F. (2016). User elicitation on single-hand microgestures. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*.
<https://doi.org/10.1145/2858036.2858589>
3. Cook, P. R. (2004, November 29). *HCI design: Princeton Real-Time Lecture notes*. Princeton University. Retrieved November 24, 2021, from <https://www.cs.princeton.edu/courses/archive/fall07/cos436/HIDDEN/RealTime/RTNotes.html>.
4. Google (Ed.). (n.d.). *Tensorflow*. TensorFlow. Retrieved November 24, 2021, from <https://www.tensorflow.org/>.
5. Gupta, R. (n.d.). *Fand : A foot interface device for four limb interactions*.  FAND. Retrieved November 25, 2021, from <http://rohitg.in/portfolio/works/fand.html>.
6. Gupta, R., & Dalvi, G. (2018). Fand: A shareable gesture based foot interface device. 2018 *IEEE 6th International Conference on Serious Games and Applications for Health (SeGAH)*. <https://doi.org/10.1109/segah.2018.8401321>
7. Huang, Y.-J., Liu, K.-Y., Lee, S.-S., & Yeh, I.-C. (2020). Evaluation of a hybrid of hand gesture and controller inputs in virtual reality. *International Journal of Human-Computer Interaction*, 37(2), 169–180.
<https://doi.org/10.1080/10447318.2020.1809248>
8. James, K. H., Humphrey, G. K., Vilis, T., Corrie, B., Baddour, R., & Goodale, M. A. (2002). “active” and “passive” learning of three-dimensional object structure within an immersive virtual reality environment. *Behavior Research Methods, Instruments, & Computers*, 34(3), 383–390.
<https://doi.org/10.3758/bf03195466>
9. McIntosh, J., Marzo, A., & Fraser, M. (2017). Sensir. *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*.
<https://doi.org/10.1145/3126594.3126604>
10. Penilla, A. R., & Taylor, A. L. (n.d.). *ASL: How to sign numbers*. dummies. Retrieved November 24, 2021, from <https://www.dummies.com/languages/american-sign-language/asl-how-to-sign-numbers/>.
11. Renotte, N. (Ed.). (2021, April 9). *Tensorflow object detection in 5 hours with Python | full course with 3 projects*. YouTube. Retrieved November 24, 2021, from <https://www.youtube.com/watch?v=yqkISICHH-U&t=9540s>.

12. Trewin, S., Laff, M., Hanson, V., & Cavender, A. (2009). Exploring visual and motor accessibility in navigating a virtual world. *ACM Transactions on Accessible Computing*, 2(2), 1–35. <https://doi.org/10.1145/1530064.1530069>
13. Ultraleap (Ed.). (n.d.). *Tracking: Leap motion controller*. Ultraleap. Retrieved November 24, 2021, from <https://www.ultraleap.com/product/leap-motion-controller/>.
14. Veit, M., Capobianco, A., & Bechmann, D. (2009). Influence of degrees of freedom's manipulation on performances during orientation tasks in virtual reality environments. *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology - VRST '09*. <https://doi.org/10.1145/1643928.1643942>
15. Way, D., & Paradiso, J. (2014). A usability user study concerning free-hand microgesture and wrist-worn sensors. *2014 11th International Conference on Wearable and Implantable Body Sensor Networks*. <https://doi.org/10.1109/bsn.2014.32>
16. Wolf, K., Naumann, A., Rohs, M., & Müller, J. (2011). A taxonomy of microinteractions: Defining microgestures based on ergonomic and scenario-dependent requirements. *Human-Computer Interaction – INTERACT 2011*, 559–575. https://doi.org/10.1007/978-3-642-23774-4_45