Re-Imagining Input: Beyond the Typewriter Keyboard

A wearable Hand Controlled Input Device (HCID) for the advancing XR revolution

Nathan Sharp

4th Year Project Report
Cognitive Science
School of Informatics
University of Edinburgh

2022

Abstract

A novel theoretical and practical work on the augmentation of human intelligence with computers (Brain-Computer Interfaces (BCI)), with a focus on input devices. We formalise a new '5AA' framework for analysing the capacities of BCIs, this drives our historical narrative of BCI 'interface revolutions' and quantifies our claim of being in the midst of a 5th fundamental revolution: the XR revolution. We identify the 'Achillies heel of XR', the absence of (anyone developing) a high-throughput, high-pervasiveness input device for XR. Tangentially, in conducting cases studies on existing, past and envisioned interface devices, we classify the many shortcomings of the (typewriter) keyboard, both for XR and more generally. Finally we begin an ongoing practical project to design and build a new Hand Controlled Input Device (HCID) to (1) better the typewriter keyboard (also mouse and touchscreen), and (2) be suitable for the XR revolution. We propose a new interface and implement it from scratch.

Keywords: *Brain-Computer Interface (BCI), Extended Reality (XR), wearables, human-computer bandwidth, ergonomics.*

Acknowledgements

To Douglas Engelbart, why has become an hero of mine. To my supervisor Antonio, who was brave enough to take on this project. To my girlfriend Xin. To my good friend Vivek, whose idea this was until he left me to become a ”billionaire” building websites. To my flatmate Timmy, who has been only slightly better than useless. To my parents, especially my father, whose advice to quit and do something else with my time has been invaluable. To the lovely staff of Edinburgh Ucreate Studio, who have bankrolled my 3D-printing compulsion. To my marker Bjoern, who has promised to increase my grade for mentioning him in my acknowledgements. To the Emacs text editor, who has tortured me so bitterly over these years. To the members of the ARTSEY discord server, who are incredible. To Sami, who set up my latex template. And to you, the reader, as I have some doubts that what follows makes much sense.

Foreword

This project was undertaken as a self-proposed 4th year Informatics project. We first developed an interest in Brain-Computer Interfaces (BCI) in the summer term of our 2nd year. Serious work begun on this project in the summer preceding 4th year, hence there is quite a lot of work for an undergraduate thesis.

This thesis was written in Org-Mode markdown.

If you are reading this copy, much of the original content has been moved to the appendix to fit the word limit. Should you wish to read this in its intended form, you can find the original in the main repository; `HCID/Thesis/thesis_XX_XX_XX.pdf` (and pick the version with the latest date).

Table of Contents

List of Figures

List of Tables

Introduction

Motivation

This project was animated by the following motivations:

- To explore the fundamental relationship between humans and computers; and its trajectory of progress.
- To better understand the future of computing, in particular XR.
- To explore how humans can be made more intelligent.
- To improve the keyboard, after suffering from its flaws of: RSI, low pervasiveness and moderate throughput.
- To best leverage our capabilities to help build a better technological future.

Summary of contributions

A summary of our major contributions:

1. A high level orientation on the magnitude of the relationship (interface) between humans (brains) and computers.
2. A practical new definition of *Brain-Computer Interfaces (BCIs)*, better representative of the continuum of extended cognition.
3. A semi-formal '5AA' framework for analysing the capacities of BCI systems.
4. A new history of BCI, informed by our 5AA framework, through our lens of 'interface revolutions'.
5. A predictive futurology of BCI, specifically with regards to the imminent XR revolution, opposing much of the prevailing wisdom.
6. The identification and analysis of the 'Achilles heel of XR': that nobody is building high-throughput, high pervasiveness, ergonomic input for XR.
7. Cases studies on existing, past and envisioned interface devices.
8. A taxonomy of the many flaws of the typewriter keyboard, including the retelling of Doug Engelbart's '3 hands problem'.

9. Insights from our ongoing practical project to design and build a new Hand Controlled Input Device (HCID) to fix (6) and (8).

Reading guide

At a high level this thesis is split intro three distinct parts, (I) Theory, (II) Case Studies, and (III) Practical. Part I, *The BCI Manifesto*, is an account of our comprehensive worldview relating to BCI.

We start with the fundamental relationship between humans and computers (§D and E) concluding, amongst other things that computers make us more intelligent (§E.1).

We go on to study interfaces (§F)– the communication bridge between humans and computers, covering their; rich interpretation under information theory (§F.2), undervalued significance (§F.1) in the literature and society, and progression in revolutionary cycles (§F.3) caused by switching costs and network effects (§F.3.1).

Next we consider Brain-Computer Interfaces (§1) as an entwined superstructure of these three elements (brains, computers and interfaces). Our definition being broader than that generally applied in the literature, we explore the philosophical underpinnings and implications (§1.3) of our world view, covering theories from the literature and extending them with our own.

This leads directly to our centrepiece 5AA framework (§2)– in which we systemise the fundamental 'Advancing Attributes' of BCI interfaces; which acts as a reference context for the rest of our work.

Next we take a detailed look at the BCI interface revolutions (§G), our historical narrative covering six revolutions that massively changed (or will change) the face of computing. From this we draw many possible lessons about what the future of computing holds such as Moore's blunder (§E.4.1) and identify the imminent XR computing revolution (§G.5).

We then consider the coming XR revolution in detail (§H), covering its features and opportunities. Specifically we review the expected output paradigm/devices (§H.1), then, looking at the potential input landscape (§H.2), we draw the key conclusion that there is an 'Achillies heel of XR': that nobody is building high-throughput, high-pervasiveness input for XR to supercede the symbolic functionality of the keyboard. We justify why this is critical, exploring the radical new opportunities (§H.2.2) of such a device.

Continuing in this interest, we collect our findings on input devices (§3), bringing together many of our discovered heuristic models, including the distinction between symbolic input and pointing devices (§3.1) and the centrality of Hand Controlled Input Devicess (HCIDs) in human interfaces (§3.2) amongst others.

Finally for part I, we cover some common misconceptions (§J) relating to our work including the iPadification fallacy which we think highly influential. Part II, *Doug Engelbart and the Keyset*, contains cases studies of past, existing and envisioned input

devices with a specific focus on the limitations of the typewriter keyboard and attempts to overcome them.

First we consider the typewriter keyboard (§I.1), noting its high-throughput value proposition in the present interface landscape and documenting the extensive flaws (§I.1.1). We also cover the interesting the two interface problem (§I.1.1.3) which explores the costs of currently having both desktops and mobile interface paradigms, rather than some unified computing environment.

We then cover ergonomic keyboards (§I.2), largely for the ways they have found innovative solutions to many of the problems of the standard keyboard.

Next we consider the computer mouse and broader catagory of pointing devices (§I.3), noting its novel value proposition and historical entwinement with the GUI computing revolution. Here we also cover one of our central theses, the 3 hands problem (§I.3.2).

We go on to briefly cover the stenotype (§I.4), a 300wpm chorded typing machine, drawing the important lesson that throughput is not even close to maximised by the keyboard. We next study the Touchscreen (§I.5) covering its advantages (§I.5.1) and drawbacks (§I.5.2) and in detail. We also conduct a short case study of the AlphaGrip controller (§I.6), the most similar historical precedent to our design specification for an new input device.

Finally we cover the flight joystick (§I.7) noting how it strongly influenced our consideration of hand ergonomics. Part III, *A wearable HCID ready for XR*, is a practical project motivated by our theoretical work and documents our engineering project to develop a new high-throughput, high-pervasiveness input device to supercede the keyboard suitable for XR.

We first present our initial reasearch (§4), including our FBD model of hand movements (§4.1) . We go on to specify the design goals (§5) and use cases (§6) of our proposed device, informed by our previous research.

Having briefly touched on our previous prototypes (§7), we cover our current prototype (§8), including the technical specification and diagrams (§8.1) and an evaluation of our progress (§8.2) such as with respect to our original design goals.

We follow with a lengthy account of our design decisions (§K), then look forward to our next prototype (§9) laying out a high level plan for evolving our design.

Finally we draw conclusions and evaluations (§10) for the thesis as a whole.

Topic Map

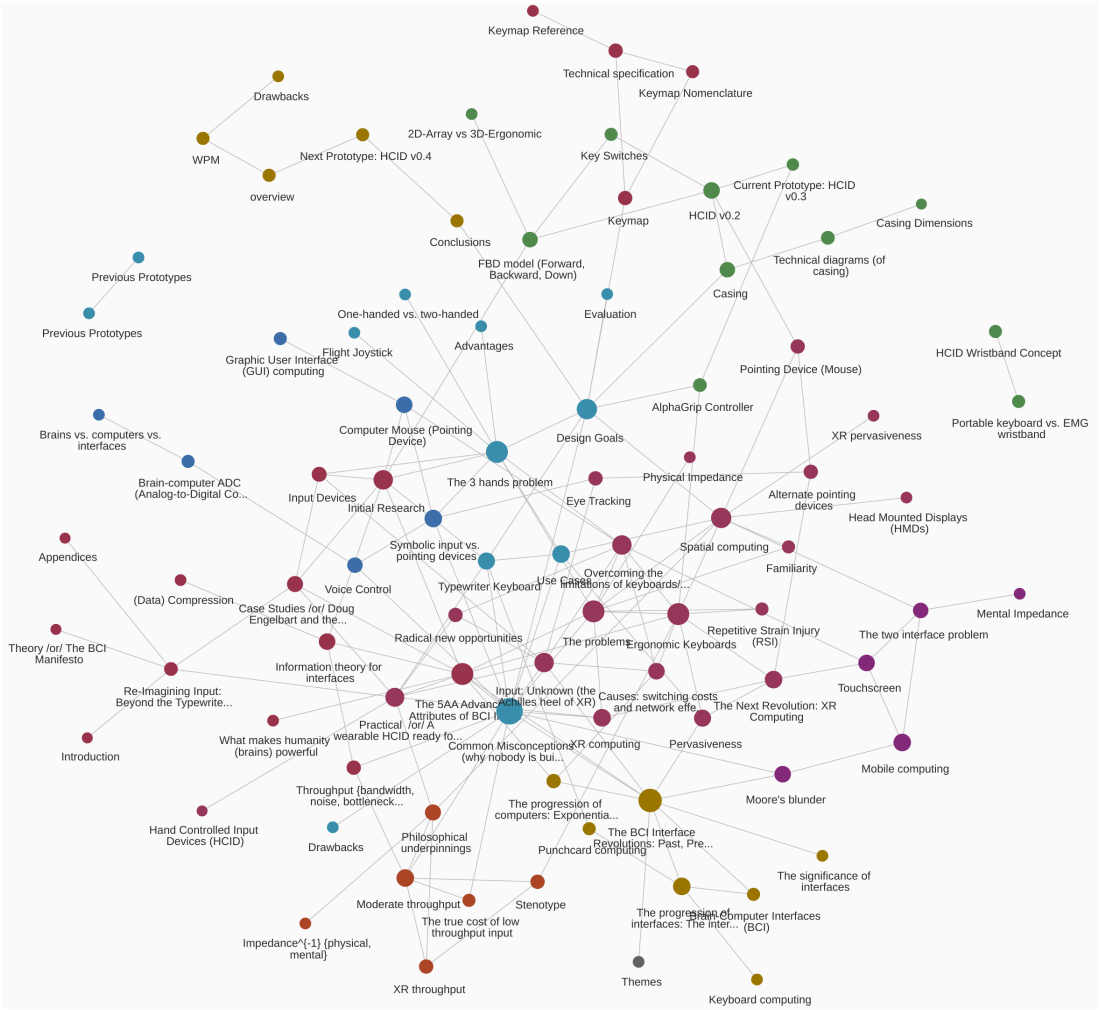

Figure 1: Map of thesis topics

Part I

Theory *or* The BCI Manifesto

Chapter 1

Brain-Computer Interfaces (BCI)

For brevity, sections on Brains (§D), Computers (§E) and Interfaces (§F) have been moved to the Appendices.

Brain-Computer Interface (BCI) A mutually specified: brain[a], computer[b] and interface[c].

[a] Human biological mind.
[b] Information processing machine.
[c] Communication structure between them.

Our use of the term *BCI* is most similar to your current conception of the term *personal computer* or just *computer*. Though we use all three terms (and also *computer interface*) somewhat interchangeably, we favour the term BCI. **This is because the term BCI helpfully encourages thinking of the human, computer and interface as a mutually specified union. Neglecting this and talking just of *computers*, fosters the hidden assumption that humans/interfaces are unimportant.** Specifically we note the widespread blindspot of assuming interfaces remain constant, when they are themselves a technological artefact, inclined towards revolutionary change (§F.3). Specifically, in our current culture, the term *computer* assumes the interface paradigms of desktop and mobile and we overlook the prospect this them changing accordingly.

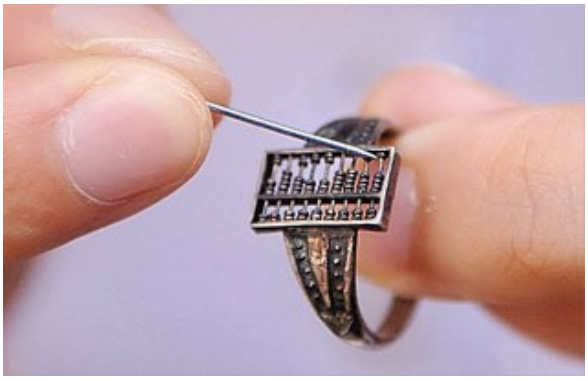

Figure 1.1: A functioning abacus ring from the Qing Dynasty (1644-1912); an early BCI?

Note, this is not how the term BCI is typically defined in the literature. **Conventionally, the term BCI it is used more narrowly to describe only interfaces that involve some sort of 'direct' electrical readings from the central nervous system; or even**

more narrowly, just the brain/outer cranium. We feel the narrower definition incorrectly implies that the most potent input signals must come from the central nervous system[1] **when we often think directly with/through our physical bodies (see *embodied cognition*).** Correspondingly, consider how the aforementioned wrong assumption violates the computational principal of *functional equivalence*; if the same input signal can be achieved from eye tracking as from electrodes in the visual cortex, is it sensible to consider the second 'True' BCI and first something other? **We put forward that the intellectually consistent position is that BCIs cover an inclusive continuum over all possible interaction channels with computers, as opposed than drawing a boundary somewhere in the skull/nervous-system.**

We do think there is a valid distinction between the conventional use of the term BCI and other input/output methods, but the distinction is purely in the biology targeted and technology leveraged rather than any deep philosophical difference. For this we suggest the new term *Nervous System Interface (NSI)*, to describe what's classically refereed to as BCI, as a sub-category of our wider definition.

1.1 Brains vs. computers vs. interfaces

Table 1.1 presents a comparative summary of attributes of brains, computers and interfaces as discussed so far.

	Brains	**Computers**	**Interfaces**
Primary Attributes	Intelligence	Functionality & efficiency	5AA
Theoretical Basis	Cognitive sciences	Theory of computation	Information theory & HCI
Analog / Digital	Analog	Digital	ADC[2]
Strength	Abstraction, planning & agency	Raw power & efficiency	Connecting brains & computers
Progress	Genetic & memetic evolution (slow)	Moore's Law (very fast)	Revolutionary cycles (fast)
Publicity	Overrated	Overrated	Underrated

Table 1.1: Comparison: brains vs. computers vs. interfaces

A key intuition we take away is: how much can you change the human? Not much. How much can you change the computer? fast but incrementally. How much can you change the interface? Massively on occasion.

[1] This in turn is based on the falsehood that mental states and processes can be found exclusively inside the brain.

[5] See Brain-computer ADC (Analog-to-Digital Conversion) (§3.5).

1.2 The inputs and the outputs

We adhere to the following convention for distinguishing the direction of information travel in BCI interfaces devices.

Input	brain → computer.	(e.g. monitor)
Output	computer → brain.	(e.g. keyboard)

This means that *input* devices primarily transmit information from the brain to the computer, whilst *output* devices do the reverse. Examples of input devices include mice, keyboards, microphones and, touchscreens; likewise, monitors, speakers, VR headsets, and printers are all output devices.

1.3 Philosophical underpinnings

In this section we tackle some of the more abstract philosophical reasoning relating to the trajectory and limits of BCIs. We'll cover some theories from the literature and extend them with of our own ideas and commentary. Principally we're interested laws governing the fundamental relationship between humans and computers. Perhaps there are insights that can structure our understanding, intuitions and design of technology.

If you find yourself doubting the absoluteness of some claim, retreat to a weaker ('mostly this is true') approximation, and note that it is most likely still producing equivalent conclusions.

The Computational Theory of Mind (CTM) The brain is a computer and cognition/consciousness is a computation, bounded by the theory of computation.

- *Note:* We regard the CMT as highly probable, though even a weaker case of the CTM (that *much* of the brains functioning is computational) suggests that the brain can be seamlessly interfaced with, and extended by, external computation.
- *Note:* The *Church-Turing-Deutsch Prinicple* implies the CTM.

The Extended Mind Thesis (EMT) The mind extends beyond the brain[a] and the body. Objects in the environment can be part of the cognitive process, extending the mind. For example: a diary, counting on fingers or written calculations. [b]

- *Note:* We think the CTM strictly implies the EMT. If the mind is computational, then like any other computer it should be possible to externalise subroutines (see also *functional equivalence*).
- *See also:* The *Enhanceable Mind Thesis*.

[a] Brain in the classical sense of the organ in the skull.
[b] For the seminal work, see [8].

Putting the previous two results together yields

⋆ **The Extended Computational Mind Thesis (The Enhanceable Mind Thesis) (ECMT)** The mind is a computer and it can extend beyond the brain/body to incorporate (what must be) external computation.

- *Intuition:* We like to think the implication of the ECMT is modifying the *Extended Mind Thesis (EMT)* into the *Enhanceable Mind Thesis (EMT)*. The insinuation is 1) the confidence of an upgrade (extend vs. enhance), and 2) the active future suffix (-ed vs -able)– progress is ours to cultivate.

The ECMT is the most potent theoretical underpinning of our work[3]. The key realisation is that the magnitude of exdended-ness is not at all fixed. It matters tremendously whether you have abacus extended mind or a AI supercomputer heads-up-display extended mind. This is a actual differential in cognitive capacity. Taking this seriously is an invitation to futher extended/enhance the mind as best as possible.

Transhumanism The practice of extending/enhancing the mind and body.

Also related to the EMT is:

Embodied Cognition The body and its interactions with the environment constitute, or contribute to, cognition. For example, in VR you can experience nausea because the input from your eyes and body does not match.

- *Key insight:* The brain (organ) is not the sole resource we have to solve problems.
- *Note:* This is closely related to the concept of *Enactivism*– that cognition arises through a dynamic interaction between the sensorimotor activity of an organism and its environment.

Functional Equivalence As a computation can be defined purely by its inputs and outputs; two systems that produce the same outputs given the same inputs are 'functionally' the same thing and should be treated as such.

- *Lemma:* Applying functional equivalence to the CTM yields two related concepts:

  Functionalism The assertion mental states can be described purely functionally.

  Substrate Independence The realisation that there is nothing special about the biological 'wetware' of the brain. It could in principle be replaced by some other computing materiel (say silicon or photons), so long as the

[3] Particularly our practical project (§III)

circuitry was functionally equivalent.

- *Note:* A key consequence of the above is that differing physical substrates of cognition would be internally indistinguishable– an alien who came to inhabit your body would have no certain way of 'feeling' what cognitive apparatus was biological and what was artificially extended.

The Very Hungry Transistor Hypothesis (VHTH) Non biological hardware will bear an increasing share of our cognition. Transistors our compete neurons for their (currently) favourable properties[a].

- *Note:* It's possible to imagine a situation in the not to distant future where the majority of our cognition is silicon extended; It might even be possible to go as far as to remove all/most biological wetware from human cognition.

[a] Such as being faster (up to 10,000,000x), more durable, and within our capabilities to engineer.

Finally, we question if/when humanity is no longer still the same species once deeply intertwined with computing:

Homo Techno An evolved species of human who's cognition is highly extended by computation.

Chapter 2

The 5AA Advancing Attributes of BCI Interfaces

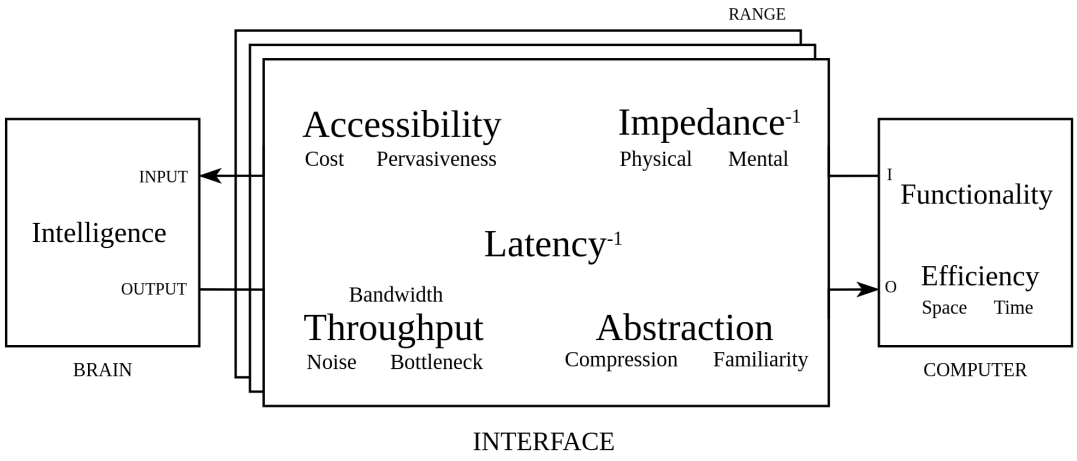

Figure 2.1: The 5AA (5 Advancing Attributes) of BCI Interfaces

The *5AA* (*5 advancing attributes of BCI interfaces*) is our fledgling theoretical framework for understanding and quantifying how BCI interfaces improve over time. They are an attempt to identify and detail the fundamental axes along which BCI interfaces progress (having have already discussed how humans approximately scale (slowly) in *intelligence* (§D.1) and computers likewise scale in *functionality* and *efficiency* (§E.4)). We propose five separate but strongly interacting advancing attributes (*5AA*) and, where applicable, their subcategories (see Figure 2.1). We try to lay out in information theoretic terms why each attribute is synonymous with an improved flow of communication (and hence greater combined system intelligence).

As you will see from our following work, we use the 5AA framework as our core explanatory tool for analysing the strengths, weaknesses and opportunities of different interfaces. Changes that advance 5AA attributes are seen as positively powerful, enough so and they can be highly revolutionary (§G). Conversely, interfaces that neglect or

underperform in certain attributes are ripe for disruption. **This is an evolutionary interpretation: better interfaces get adopted over time.**

Note that many of the 5AA's concepts have been lifted direct from our study of information theory (§F.2). For example, *availability* is a direct analog to *system uptime*. Additionally, though measuring attributes quantitatively is possible, we tend to stick to broad comparative measures e.g. "Interface X can do everything interface Y can do but in E additional useful environment, hence making X more pervasive."

Note also the previously defined concept of *range* or *frequency band*, which bounds our 5AA analysis over a spectrum of communication, "Allowing a system to be decomposed into many discrete channels in which the bandwidth/noise/etc. may be treated differently"; say, over some category of symbols (e.g. the ones found on a standard keyboard)[1]. For example a monitor might have great visual throughput but no working audio, and a keyboard might have high WPM on prose text but be slow to type symbols. The concept of range allows us to make this distinction.

Finally, also consider that the 'advancement' is *local* not *global*. For example cost decreases when holding other attributes constant, but its possible for the cost of the average computer in increase whilst also becoming more powerful.

2.1 Abstraction {compression, familiarity}

BCI interfaces tend towards increasing *abstraction* over time. This is composed of *(data) compression* and *familiarity*.

> **Abstraction** Arranging complexity so the human and computer deal with fewer, higher-level inputs/outputs.

Abstraction is about allowing the user to do 'more with less'. For example, WYSIWYG text editors *compresses* a lot of the underlying complexity of document processing in a simple interface with much *familiarity* from physical paper. Our concept of abstraction covers the many complexities of UX design (e.g subtleties of symmetry, proximity, etc.). Note that Abstraction is closely related to the attribute of (decreasing) *mental impedance* as better abstractions require less cognition to utilise.

2.1.1 (Data) Compression

Our definition of compression is lifted directly from information theory:

> **(Data) Compression** The process of encoding information using fewer bits that the original representation.

For example, modern GUI operating systems connect to a WiFi network in a couple of clicks (or better none at all), this could previously be quite long process with

[1] If you wanted to make serious quantitative measures, you would probably want to run experiments sampling from a representative corpus of text of know symbol frequency distributions.

many commands in the terminal. Note that data compression can be *leaky* (*lossy*)– disregarding some of the original information.

2.1.2 Familiarity

Familiarity The extent to which an interface provides an isomorphism to some already known representation.

Familiarity can be viewed as a 5AA-compression *across* interface domains– if some abstraction is already known by a user, it is optimal to use it rather than create some new equivalent. Familiarity leverages network effects as designers 'cluster' around certain common interaction models[2] For example the (top of screen) drop-down menu-bar became a familiar navigation abstraction across many GUI computer applications; there was considerable backlash when Microsoft Word removed one in 2000. See also Spatial Computing (§H.3), which leverages specifically, the familiarity from the physical world.

2.2 Throughput {bandwidth, noise, bottleneck}

BCI interfaces tend towards increasing *throughput* overtime. *Bandwidth*, *noise* (including *errors*) and *bottlenecks* all contribute to throughput.

Throughput The real average rate of successful transfer (over some range). As a simplification we will principally be interested in Words Per Minute (WPM).

Note that the throughput depends greatly on the *range* of the interface/task. For definitions of *bandwidth*, *noise*, and *bottleneck* see Information theory for interfaces (§F.2).

2.3 Latency^{-1}

BCI interfaces tend towards decreasing *latency* over time.

Latency (Lag) Time delay sending a message between a human and a computer.

Latency is arguably the most perfected attribute of modern computer interfaces: which feel essentially instantaneous to humans[3]. This is partly because digital computers now operate much faster than human 'framerate', completing many thousands of operations in the blink of an eye. Note, in the early days of computing, latency was a huge bottleneck. For instance, having a printer as the primary output device, rather than a monitor, put output latency in the seconds/minutes range, greatly limiting human-computer communication.

[2] Hence incurring switching costs and contributing to the interface revolution cycle.

[3] Should computers ever become sentient, we can expect human latency to spell the end for us.

Latency as an issue has the potential to reemerge in new (resource intensive) computer paradigms such as VR, or with respect to the need for constant connectivity, as with the internet.

2.4 Impedance^{-1} {physical, mental}

BCI interfaces tend towards decreasing *impedance* (or *load*) over time. Impedance in humans can be *mental* or *physical*.

Impedance (load) A measure of the resources required to transmit/receive a message over an interface.

Impedance^{-1} (Admittance) How easily an interface will allow a human/computer to use its functionality.

Decreasing impedance is essentially a generalisation of *ease of use* and is closely related to the attribute of *abstraction*, as powerful abstractions reduce impedance. In humans we split impedance into physical and mental categories, however note that this distinction is not clear cut (see *embodied cognition* in Philosophical Underpinnings §1.3).

2.4.1 Physical Impedance

Physical impedance (physical load) A measure of the physical resources required to transmit/receive a message over an interface.

Physical impedance in humans measures the extend to which the body is consumed in using an interface; including subtleties such as disorientation and fatigue. Physical impedance incorporates much of the important topic of *ergonomics*. Generally we can think of good ergonomics as reducing physical impedance, for example, switching from a keyboard that causes RSI to one that does not. Physical impedance also relates strongly to *pervasiveness* in that lower physical impedance can often mean increased pervasiveness (the less demanding an interface is to carry, the more places you can take it). For example, a keyboard that takes two hands to operate has greater physical impedance (and is hence less pervasive) than a keyboard that only requires one.

2.4.2 Mental Impedance

Mental impedance (Cognitive Load) A measure of the cognitive resources required to transmit/receive a message over an interface.

Mental impedance or *Cognitive Load* is an abstract concept covering all the complex ways using an interface can take up more or less of your mental capacity, including; concentration, switching/learning costs, mental congestion, noise, sensory attenuation,

etc. As you can imagine, mental impedance is highly related to the concept of *5AA-abstraction*.

2.5 Accessibility {cost, pervasiveness}

Interfaces tend towards increasing *accessibility* over time; specifically decreasing in *cost and increasing in /pervasiveness*.

> **Accessibility** A measure of the opportunities/accessibility for a human to use a computer interface.

Increasing accessibility essentially means 1) more people use computers– as *cost* decreases, in 2) more varied situations– as *pervasiveness* increases. Notice that our concept of accesibility has been derive from the information theory primitive of *system uptime*.

2.5.1 Cost

We'll leave defining cost as a an exercise for the reader. Note that cost was an important factor in the ascendance of mobile over desktop and likewise in Microsoft's initial domination of the PC market with an otherwise inferior product.

2.5.2 Pervasiveness

> **Pervasiveness** The omnipresence of a computing environment; how much of the time or in how many situations a user is meaningfully access to use a computer interface.

Figure 2.2: Computational physicist Stephen Wolfram experimenting with (post-desktop pre-XR) pervasive computing environments. From left: treadmill, laptop strap, standing desk [53].

The best demonstration of pervasiveness by sample: a laptop is more pervasive that an desktop, a smartphone is more pervasive than a laptop, a smart watch is almost entirely pervasive, a heads-up display (think AR glasses) is more pervasive yet[4]– as

[4]Note that AR is more pervasive than VR as there are lots of situations you can't take the VR head cave.

you don't even have to position your wrist to interact, and a neural implant interface is the pervasiveness holy grail. Increases in pervasiveness are some of the most powerful changes to computer interfaces being highly capable of causing an interface revolution (§G). **Pervasiveness has been a major driver in five of the six interface revolutions (§G) we identify and we think the coming XR computing revolution (§H) will be the same.**

Chapter 3

Input Devices

Having identified input as the achillies heel of XR, we dive deeper into our theoretical study of BCI input devices, with a focus on XR. For case studies see §I and for our practical research see initial research (§4).

3.1 Symbolic input vs. pointing devices

One valuable heuristic our research revealed is the distinction between *symbolic input (devices)* and *pointing devices*.

> **Symbolic Input (Device)** Input of symbolic information (text, symbols, commands, etc.) to a computer. The canonical symbolic input device is the keyboard.

> **Pointing Device** Input device that controls a continuous spatial pointer, approximately representative of *attention*; generally also equipped with *inspection* (click) functionality.

Note that this paradigm distinction closely maps on to the keyboard-operated-desktop vs. touchscreen-operated-mobile divide.

Symbolic input and pointing devices represent genuinely distinct modalities of how we interact with computer. We suspect comes from the evolved cognitive psychology of having two distinct modalities of human interaction– pointing-at and investigating/exploring things vs. manipulating things you already know. **The important realisation is that each has their relative strengths and weaknesses so a universal interface paradigm should leverage both, rather than have be dominated by one.** A helpful comparative summary between the two is provided in Table 3.1.

	Pointing Device	**Symbolic Input**
Primitive	Attention & inspection	Commands & complexity

Continued on next page

Continued from previous page

	Pointing Device	Symbolic Input
Domain	Unfamiliar exploration	Familiar manipulation
Benefits	Ease of use & discoverability	Speed & functionality
Data	Continuous & spatial	Discrete
Examples	Mouse, touchscreen & eye tracking	Keyboard & buttons
Paradigm	Mobile	Desktop

Table 3.1: Comparison: symbolic input vs. pointing devices

For an example of relative benefits and drawbacks, interfaces designed for pointing devices can be highly intuitive as functionality is visually discoverable (cannot be a 'hidden' secret); however, you can only go as fast as you can control a pointer over predefined commands. Conversely, symbolic input can operate very quickly[1] and is designed to deal with a greater, more complex range of input and abstraction. A good example of this is programming– there are far too many keywords and variable names to select them from a GUI, and you often want to create (symbolic) macros to bundle together complex specialist functionality. Also note that (graphic) interfaces for pointing devices are more individualised and generally more intensive to build than APIs for symbolic interfaces. Hence there are many situations that require (or first provide) a symbolic interface, such as fixing the internals of a computer or a cutting edge software technology.

Many of the most powerful interfaces combine the best qualities of both. Common is an expansive GUI interface with many shortcuts– as seen in professional desktop applications such as IDEs and image editors. Even so, synergy is currently hamstrung by the 3 hands problem (§I.3.2).

We notice the tendency of those building XR computing to drastically underestimate the value proposition of symbolic input. This is one face of the broader *iPadification fallacy*, responsible for the in vogue attempt to build XR computers devoid of serious symbolic input, instead with only mobile-esque pointing devices.

See also Voice Control (§3.4) for a discussion on a potential future third major computer interface modality (linguistic).

3.2 Hand Controlled Input Devices (HCID)

Hand Controled Input Device (HCID) Thesis The most sophisticated tools humans build tend to be hand[a] controlled.

- *Note:* Throughout the rest of this work we often use the term *HCID* as a synonym for 'dominant input device to a computer'.

[1] At the press of a button already under the finger.

[a] And by extension including the arm.

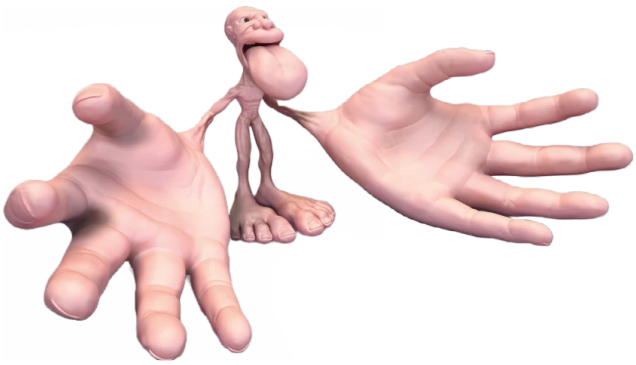

Figure 3.1: The Motor Homunculus: The body scaled by the brains motor control real estate

Examples of HCID thesis include blacksmiths, sculptors, artists, surgeons, military combatants, musicians and now programmers. Figure 3.1 alludes strongly to the biomechanical/neurological/evolutionary underpinning of this reality.

Perhaps a corollary of the HCID thesis is that input devices should strive to use the whole hand (say, not just the thumb and index finger as with smartphone keyboards), and do so in harmony with the natural ergonomics of the hand.

The HCID thesis raises a challenge of legitimacy for input devices that wish to buck this trend, some of which we shall later discuss. For example, the vision of an NSI as a dominant computing paradigm of future rests on the supposed fluidity of being able to control a computer with just the mind. One possible interpretation of the HCID thesis, inspired by embodied cognition, might be that our control of complex tools, such as a computer, cannot always be better 'simulated' in the mind, and is rather best perfected in *haptic thinking*– through movement and manipulation of the hands.

3.3 Eye Tracking

Briefly we would like to make the cases for eye-tracking as an input device for XR and the future more broadly, specifically as a pointing device.

Firstly note that eye tracking hardware is well suited to XR as you already have considerable computing apparatus strapped to your face. Our fundamental argument is that we thing eye tracking is the ideal pointing device (§3.1), an instantaneous and effortless demarcator of attention. For every other pointing device, you necessarily already move your gaze over the desired path; eye tracking is just cutting out the middle man of hand movement and all impedance that comes as baggage.

Consider also the widely accepted evolutionary argument of *the cooperative eye hy-*

pothesis, that humans developed white sclera[2] (outer ring of eye) to make it easier for humans to follow each others gaze when communication and cooperating. This suggests that we are already highly evolved to use eye gaze as a pointing device of attention.

We think there is a high chance that eye-tracking replaces the mouse/trackpad/joystick as the primary indicator of attention and inspection pointing device in XR interfaces[3] As for the question of feasibility, we have good reason to believe eye tracking can be harnessed accurately [34, 43] and confined within a reasonably sized headset [48].

3.4 Voice Control

A common misconception is that there will soon be little need for traditional (symbolic/pointing) computing interfaces as everything will soon be voice controlled. Whilst voice assistants will likely continue to improve and soon take on a larger role in human-computer interaction, we think generally this is a misunderstanding of how most people use, and will continue to use computers.

In summary, we don't see voice interfaces as being capable of replacing symbolic input and pointing devices (§3.1) as voice control can neither well replicate their functionality nor displace the need for it. **Humans understand and communicate in verbal-linguistic, symbolic and visuospatial ways[4], but voice control interfaces only provide a strong channel for verbal linguistic thinking.**

Perhaps instead, voice will add a third, higher level, mode of input *linguistic*– for dealing with large, abstract, yet typical tasks, such as making calendar appointments, searching the web, or parking a car; particularly useful in hands-free environments.

3.4.1 Advantages

Voice control does offer advantages, chiefly the high abstraction of natural language. Natural language is already universally familiar and can be very terse in specifying complex, high level functionality, such as 'Paint me a self portrait in the style of Henry Raeburn' [11].

3.4.2 Disadvantages

However there are the limitations of voice control, in consequence of which it should not serve as the sole, nor principal, form of input to computers.

First is the problem of discoverability, with a voice interface it is not easy to know what commands are available, especially when faced with a lot of functionality.

Second is the difficulty of operation in a noisy place, and the rudeness of being in a quiet, yet occupied one. This leads to really unreliable pervasiveness.

[2] As opposed to all our nearest animal relations who have consistently dark eyes.
[3] Freeing hand control for manipulation as per the HCID thesis.
[4] Among others.

Thirdly there is the issue of low throughput for certain tasks. Generally everything requires a full sentence or specified keywords; no rapid set of keyboard shortcuts. For example, consider working on a spreadsheet (something we find hard to believe humans will relinquish the need for). For many a primitive spreadsheet task, voice is considerably more cumbersome than the status quo, e.g. scrolling around, making selections, exploring menus or writing formulas. This point is generalised in that that humans communicate with computers in visuospatial, symbolic, as well as linguistic ways.

Finally there is the more abstract issue of problems relating to brain-computer ADC (§3.5). In summary, it is not clear that a computer can, or should, convert natural language to computer commands (over a large input range) considering the complexity, ambiguity and situational dependence of speech.

3.5 Brain-computer ADC (Analog-to-Digital Conversion)

Brain-Computer ADC (Analog-to-Digitial Conversion) That BCI interfaces must bridge the complex analog–digital divide between humans and computers.

We have a theory that communicating between humans and computers is a hard problem fundamentally because humans are analog and computers (currently) are digital. For computers to best utilise communication over an interface, there must be little ambiguity in the communication. For humans to use an interface, it must have some isomorphism to their familiar environment– e.g. natural language, color or 3D geometry. This means complexities and tradeoffs and leaky abstractions in the communication interface between them.

We argue that the best interface devices are ones that provide good abstractions for both the human and the computer to meet in the middle– rather than forcing the one to go all the way to the others paradigm. In other words, it helps if a human can make their input more rigorous/unambiguous; both for the direct reason of better input to the computer, but also because it brings the human closer to the computers internal model, guiding better use. Similarly it helps if a computer can represent its processes in high level human abstractions such as GUIs, rather than say, strings of binary.

Lets take some illustrated examples through this lens. Punchcards are poor interface devices in this framework because the ADC is done almost entirely in the human, who has to learn to communicate in slow and conceptually unnatural punched holes (yet this is a pretty direct form of input for a computer). Keyboards are good interface devices because keyboards are only a short step from natural language, yet a computer can easily decode this into a number based system (ISO keycodes) as each letter is serialised into a discrete button press. The computer mouse is similarly strong ADC system, control feels naturally continuous to the human but is easily serialised by the computer to a discrete point in a 2D grid. Eye tracking is a little more complex as you need some considerable computer side ADC to transform eye saccades into stabilised discrete movements, but, at least from the human side, the intention of discreteness

(selecting and item from an interface) is already there. Voice control is currently a poor interface under this interpretation because ADC is almost entirely done my the computer from a very complex and noisy analog domain, suggesting this might not be a good idea. NSI is also a tricky interface device in this framework. For ADC of analog neural patterns, the computer must solve some complex classification problem over some boundary that might be highly fuzzy. The hope is that there are sufficient pockets where the human can indicate discrete intention and the computer can register this.

3.6 The 3 hands problem

See The 3 hands problem (§I.3.2).

Part II

Case Studies *or* Doug Engelbart and the Keyset

For brevity, all case studies (§I) have been moved to the appendices.

Part III

Practical *or* A wearable HCID ready for XR

Summarising our most important work so far, we 1) Developed the The 5AA Framework (§ 2) for evaluating the strengths and weaknesses of different interface devices, 2) Established that contemporary input devices are not suitable for the coming XR Revolution (§ G.5), fundamentally because they lack pervasiveness (typewriter keyboard), and/or bandwidth (VR controller), 3) Argued that text/symbolic input is essential to general-purpose computing and it is a common misconception to overlook this., 4) Postulated that Hand Controlled Input Devices (HCID) are, and will remain, the predominant input mechanism[5] for BCIs (§ 3.2), 5) Identified typewriter keyboard as a leading interface bottleneck in the pervasiveness of BCIs (§ I.1) and additionally identified several other important ways they could be improved, and 6) Conducted Case Studies (§ I) on existing, past and envisioned HCID solutions. We now task ourselves with practically innovating in this space.

Goal design a high-bandwidth, high pervasiveness HCID to supersede the keyboard and mouse appropriate for XR.

A diverse range of knowledge and skills had to be acquired to undertake this project, much of which we had to learn from scratch.

Specifically:

- *Software.* Micro-controller programming, specifically the subfield of keyboard firmware.
- *Electronics.* Basic electronics including the reading and creating of electrical diagrams and PCB design.
- *Design Engineering.* CAD modelling, 3D printing and material science.
- *Ergonomics.* Including the study of hand mechanics and RSI.
- *Cognitive Science.* Human centred keymap design and abstractions.

[5] certainly for text/symbolic input.

Chapter 4

Initial Research

We researched various **theoretic** aspects of input devices. Influential to our thinking was Information Theory (§ F.2), the Extended Mind Thesis [8], and Doug Engelbart's early work on augmenting human intellect [13]. They influenced the development of our theoretical framework, The 5AA (§ 2). We also researched the **history** of input devices in depth, most notably the focusing on (1) the strengths and limitations of the Typewriter keyboard (§ I.1) and (2) alternate input devices, naturally starting with when Doug Engelbart invented The Mouse (§ I.3) and inadvertently created The 3 hands problem (§ I.3.2), leading him to propose *the keyset* as the first potential solution. We collected **case studies** (§ I) of attempts to build an alternative input devices to the typewriter keyboard. The themes found, such as chorded-input, directed our ideas for hardware, firmware and keymaps. Finally, we studied **hand biomechanics**, attempting to engage in *first principles thinking* with regards to what movements a hand makes most naturally so as to best design a HCID around this. This lead us to devise our FBD model of finger movements (§ 4.1).

4.1 FBD model (Forward, Backward, Down)

The FBD model is a simplified model/mapping of finger movements we devised with a handheld button-based environment in mind. In this model fingers can move *Forward*, *Backward* and *Down* (see fig. 4.1) and was born out of careful consideration of how hands could be expected to move with regards to pressing buttons within its grasp.

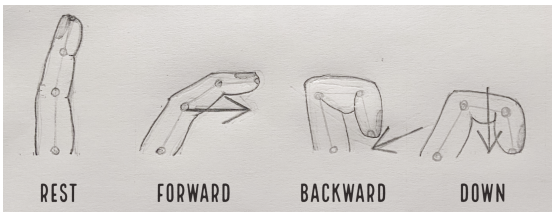

Figure 4.1: FBD model of finger movements. The arrow indicates the force vector in each case.

Chapter 5

Design Goals

A list of *aspiring* design goals for a new HCID, generated from our research so far.

1. **Hand controlled** a Hand Controlled Input Device (HCID).
   (a) **(Optionally) One Handed** solving The 3 hands problem (§ I.3.2).
2. **Pervasive** operable as 'everywhere' as possible, even more so than mobile.
3. **Functional** able to output[1]: the full range of keyboard symbols, mouse controls, letter combinations, words, phrases, autocomplete, gestures, game-controller commands, 6DOF movement, etc.
4. **Ergonomic** feel natural and pleasing for the hand to use (and should not cause RSI).
   (a) **Tactile (touch typing)** be operable without looking (unlike smartphone touchscreen keyboards). Note, standard keyboards are not fully tactile, although it is possible to learn to touch type (without looking).
5. **Fast** have bandwidth as fast or faster than a keyboard and mouse for all tasks. In the limit, it could be as fast as thought ≈300wpm, (or speech ≈200wpm).
6. **Spatial** making use of movement and gestures natural to humans and usable from a flexible range of hand/arm positions (see spatial computing; § H.3).
7. **Isomorphic** leverage familiarity of known tools. For example, providing access to a keymap similar to QWERTY. would make the device easy to learn.
8. **Extensible** it should allow for future growth and be easy for users to modify/extend.

These criteria will not all be fully reachable within the scope of this dissertation, however, I do believe they will eventually all be a necessity.

[1] in rough order of importance.

Chapter 6

Use Cases

The following are some of the important use cases to consider for our new HCID device.

Pervasive Use There's a whole class of use cases downstream from a high-bandwidth input device pervasive enough to be used anywhere. Think giving a presentation whilst walking and typing, writing an essay lying comfortably[1] in bed, taking notes in a meeting whilst going for a walk; all with effortless access to a desktop computing environment through high-bandwidth input. **This pervasiveness will allow users to escape the sedentary lifestyle of sitting at a desk all day.** The comparison to mobile devices here is important as they are a halfway house to much of this functionality, but lack high bandwidth input and 'ergonomic' touch typing. See also the main section on Pervasiveness (§ 2.5.2).

XR (VR, AR, etc.) Related to *pervasive use*, a compelling and arguably irreplaceable use case for our HCID would be in XR; a platform many are betting on being the future of computing. Keyboards are obviously unsuitable here as they force you to sacrifice freely moving your arms or body. **There is no existing solution for moving text input in XR beyond point-and-click on a virtual keyboard**[2] (see Figure 6.1), which almost certainly does not have the bandwidth capacity for general-purpose text/symbolic input. This use case is grounded in a vision for XR as a platform for general-purpose computing. This is in stark contrast to the vision currently pursued by (at the time of writing) the world's most influential entity in this space, Meta Inc. [51], led by CEO Mark Zuckerberg, whose current focus seems to be on XR as a social-interactive gaming/entertainment 'Metaverse'. Meta's framing effect contributes to our understanding of why XR as a general-purpose computing platform is underrated.

Preventing Repetitive Strain Injury (RSI) Many users of traditional keyboards suffer

[1] no, using a laptop in bed is not naturally comfortable.

[2] may I suggest a "swipe" keyboard (CITE) as a substantial improvement. See also; [BROKEN LINK: ee9403cf-caa9-4884-a4d5-b52cb8d4c8a3] (§ [BROKEN LINK: ee9403cf-caa9-4884-a4d5-b52cb8d4c8a3]) for makeshift partial solution.

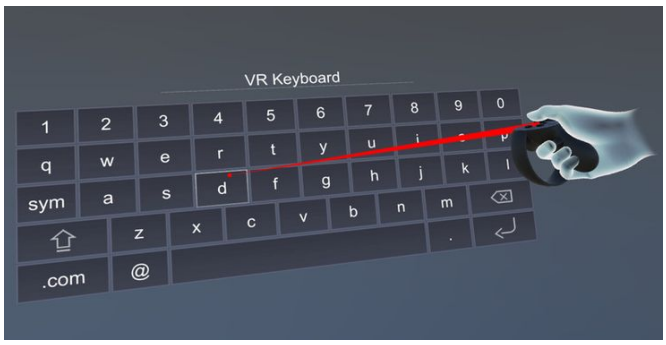

Figure 6.1: A state-of-the-art VR on-screen point-and-click keyboard. This is obviously a terrible long term solution for general purpose use.

from RSI[3]. Effective solutions do seem to exist in the form of ergonomic keyboards and alternate layouts such as DVORAK. Any new device should incorporate the existing wisdom on how to avoid users developing RSI.

One handed input There are many reasons why a user may only have, or want to have, one hand available, the opposite of which is assumed for efficient use of a typewriter keyboard. For example, some user may have an injury or disability and many professionals need to constantly use a mouse (or some other tool) with the other hand (see The 3 hands problem; § I.3.2). There is also the more open advantage of (having the option to) have a hand free whilst maintaining full access to a computing environment, adding flexibility that makes usage practical in many new hybrid (pervasive) scenarios. For example, a shopkeeper who still needs a hand free to perform manual tasks.

Spatial computing A new input device could incorporates natural movement of the body, such as gestures, to power fluid and intuitive input (see Spatial computing; § 2.5.2).

Productivity There are many ways a new HCID could be adopted as an indispensable productivity device. For example, if progress was made on throughput allowing input text/commands closer to the speed of thought there would be obvious productivity advantages. Similarly, if the The 3 hands problem (§ I.3.2) could be overcome, allowing true simultaneous use of a keyboard and mouse, it would be a strong productivity proposition for existing professionals who need to use both a mouse and keyboard. Additionally, many people may be attracted to VR/XR for the productivity benefits of limitless screen real estate, and you would imagine they would want an input device native to the spatially fluid form-factor.

[3]One study suggested that 60% of IT professionals suffer symptoms of RSI at some point [28].

Chapter 7

Previous Prototypes

All elements of the design including the casing, keymap and electronics went through radical design iterations to reach their current state. Write-ups on the most important previous prototypes can be found in Previous Prototypes (Appendix B).

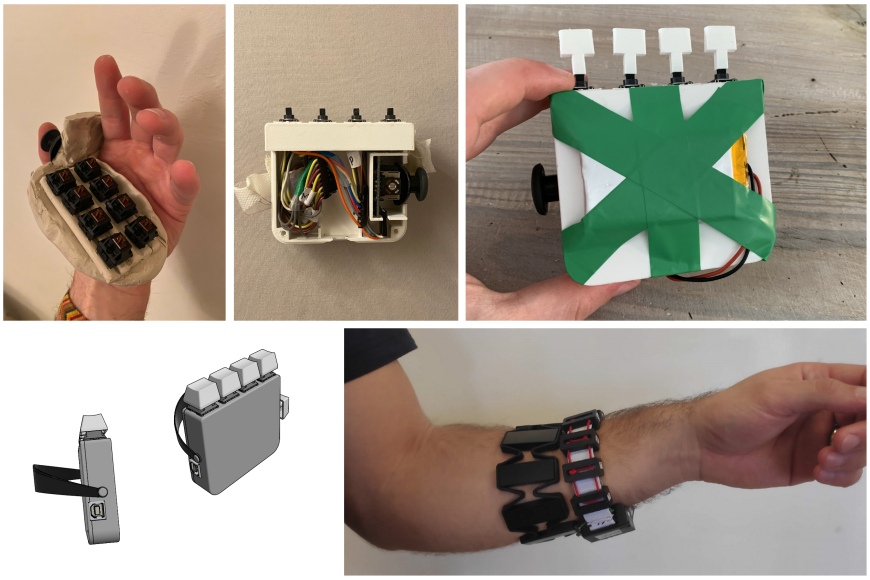

Figure 7.1: Previous prototypes and ideas

Chapter 8

Current Prototype: HCID v0.3

Figure 8.1: HCID v0.3: Using in hand

HCID v0.3 is a fully operational early prototype of a **one-handed ergonomic handheld input device functional and fast enough to fully replace a mouse and keyboard**. Although functional, HCID v0.3 is still far from being a consumer product, we still intend to iterate on major design elements such as the button layout, there are many flaws to be fixed and much of the hardware is still makeshift such as the handstrap or hand-wired electronics. This is understandable given the time and resource constraints of the project and we have many straightforward improvements suggested in Next Prototype: HCID v0.4 (§ 9).

In overview, HCID v0.3 consists of an ergonomic 3D-printed casing with buttons on two faces: the *front* (thumb-controlled) and *back* (finger-controlled). The back has a 2×4 array of standard mechanical key switches and the front has a trackball, two additional keys[1], a `Pause` button lock the input[2] and a small OLED screen for displaying information such as the battery, layer and modifier status. It is fully usable with both

[1] six were originally planned.

[2] not fully implemented at time of writing

just one hand and two hands– in the latter, you get two identical mirrored keymaps (sharing modifier keys). See also design decisions and discussion (§K).

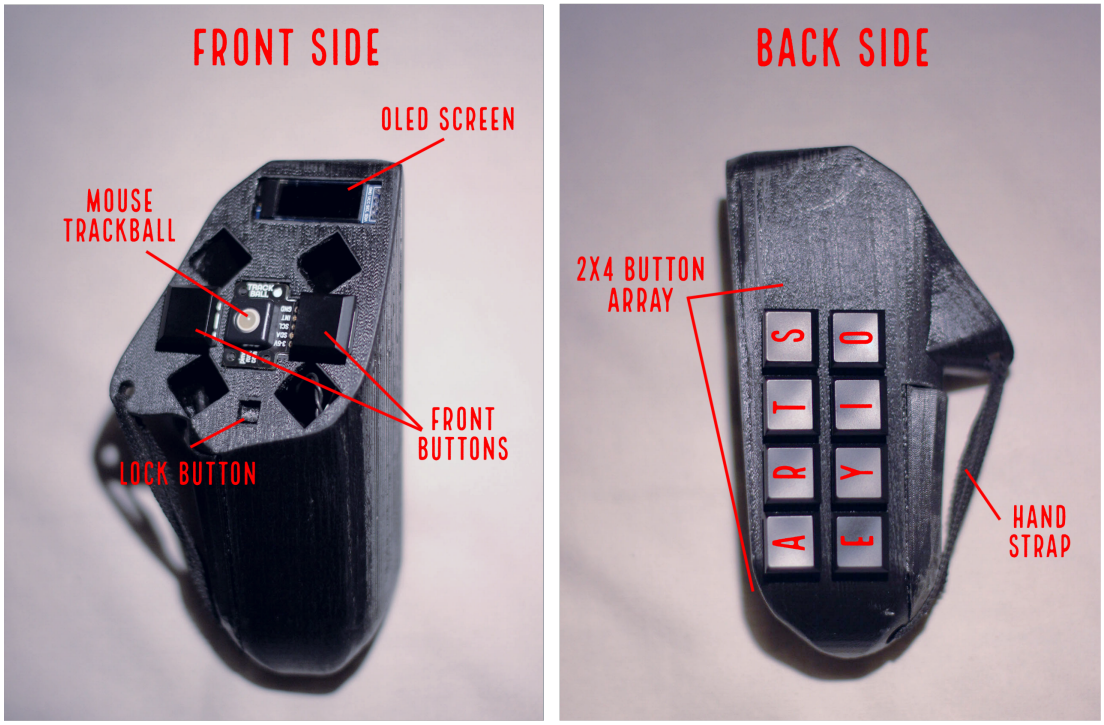

Figure 8.2: HCID v0.3: Annotated views

To output the full range of keyboard and mouse symbols with relatively few keys, HCID v0.3 makes extensive use of *chords* and *layers*. In measuring performance, we have been able to consistently achieve one handed typing speeds of over 30wpm (see WPM; § 8.2.1) with quite limited practice, and believe that 40-50wpm is easily possible.

8.1 Technical specification

8.1.1 Bill of materials

See Table 8.1. Note that access was needed to a soldering iron and 3D printer. Looking to decrease costs in future, most items would be cheaper in bulk. Research suggests that the costs could be driven down to around £50/item at 100 units. Note that this is only the cost for a single-handed device.

8.1.2 Technical diagrams (of casing)

The 3D printed casing was designed using CAD (Autodesk Inventor [17]). All full size technical drawings can be found in Appendix C.1. An example is shown in Figure 8.3. Source files can be found in the `/CAD` folder of the main repository [39]. For discussion see Design Decisions: Casing (§ K.8).

Part	Details	Quantity	Total Cost (£)
Microcontroller	Nice!nano v2 [29]	1	22.50
LiPo battery	3.7V 1200mAh [15]	1	7.21
OLED Screen	128 × 32 [2]	1	2.50
Mouse trackball	Pimoroni [44]	1	13.8
Key switches	Kailh low profile [24]	10	11.00
Key caps	[24]	10	7.90
Misc. electronics	Wire, solder, etc.	1	0
Casing	3D printed	1	0
Screws	3mm × 25mm	1	0
Hand strap	Homemade	1	0
Total			57.70

∗ Total cost excludes shipping (roughly an additional £10)
∗∗ Where cost is zero, materials were provided by the university

Table 8.1: HCID v0.3: Bill of materials

8.1.3 Wiring schematic

An electronic schematic diagram is provided (Figure 8.4). It was not strictly necessary for this prototype as it was hand-wired, but it does aid clarity and was partly produced in preparation for the next prototype where we intend to print a custom PCB (which compels a formal wiring schematic).

8.1.4 Keymap (code)

For any unfamiliar terminology, particularly that in *italics*, see Keymap Nomenclature (Appendix A). The *back* (fingers) dominate the symbolic input (letters, numbers, punctuation, functions) whilst the *front* (thumb) controls the pointing device (mouse), two additional general-purpose keys (currently set to `SHIFT` and `GUI`) and a `Pause` button to lock the input from accidentally pressing buttons when you don't want to use the device. The *back* keymap is a modified version of that created by the `ARTSEY` keyboard community [6].

The keymap (back) is best understood as a reference diagram which can be found in Keymap Reference (Appendix C.2). In summary, the keymap has a *base layer* containing *alphas* (letters), *essentials* (`Space`, `Return`, `Tab` and `Escape`), *modifiers*[3] and a subset of *symbols*; all of which are achieved through 1 to 4 finger chords. These keys are laid out (in order of priority) to 1) have the most common keys in the easiest to reach positions, 2) avoid the most straining movements, and 3) have an isomorphism to symbol shape. For an example of (1), the 8 base keys (keys that function with a single finger press as on a standard keyboard) are `A-R-T-S-E-Y-I-O`, approximately the most common letters in English. For an example of (2), multi-finger chords that involve the pinky and middle finger have been kept to a minimum. For an example of (3), the

[3] implemented as *Sticky-Keys*

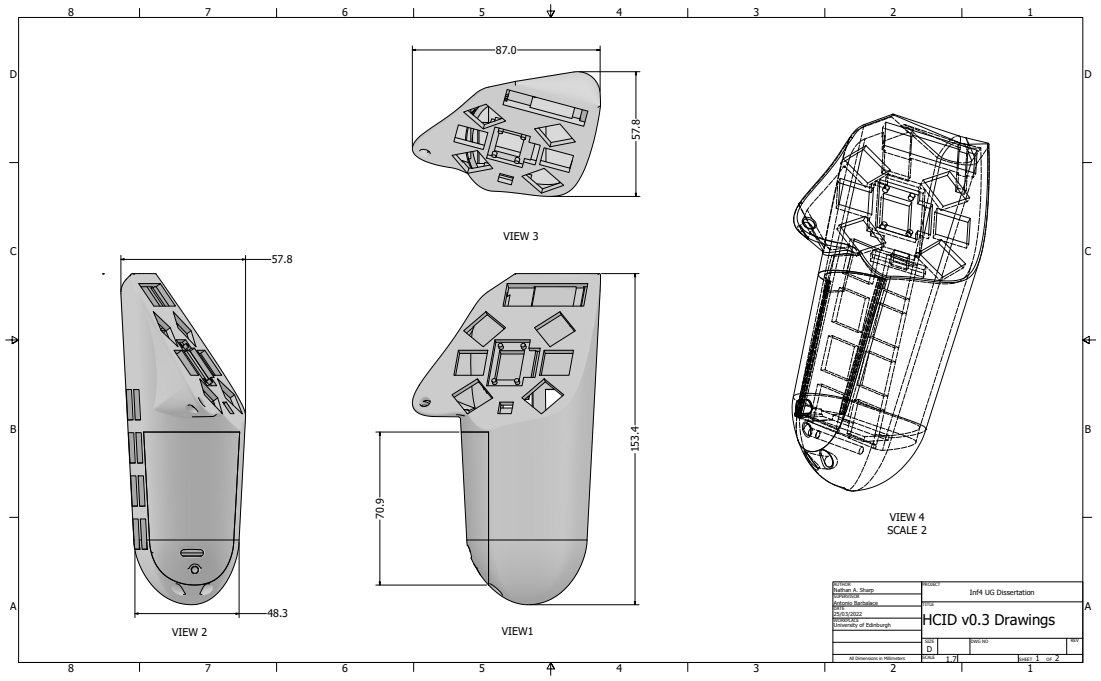

Figure 8.3: HCID v0.3: Technical diagrams: Selected example

letter `l` has the chord ■ ■ ■ □ (left hand), which looks like the letter making it feel natural and easier to remember. Beyond the base layer, the remaining functionality (*numbers*, *symbols*, etc.) is achieved through zmk's [55] *Hold-tap Momentary-Layers*– several *Layers* that are activated whilst holding down some button (generally a corner of the 2×4 array). For more discussion on the keymap design, see Design Decisions: Keymap (§ K.11).

The custom firmware for this keymap can be found in the `/zmk` folder of the main repository [39].

8.2 Evaluation

8.2.1 WPM

We have been able to consistently achieve typing speeds of over 30wpm (see Figure 8.5), with quite limited practice, and are confident that 40-50wpm is easily possible with more practice. Note this is using the device one-handed, we haven't tried two-handed yet as we have only built left-handed prototypes, but believe it should be 1.3-1.6$\times$ faster.

Anecdotally, whilst 30wpm is reasonably comfortable for daily use[4], it can be frustrating for activities that require extensive use of *symbols* and *modifiers* such as programming.

[4]The author types around 60wpm on a standard QWERTY layout.

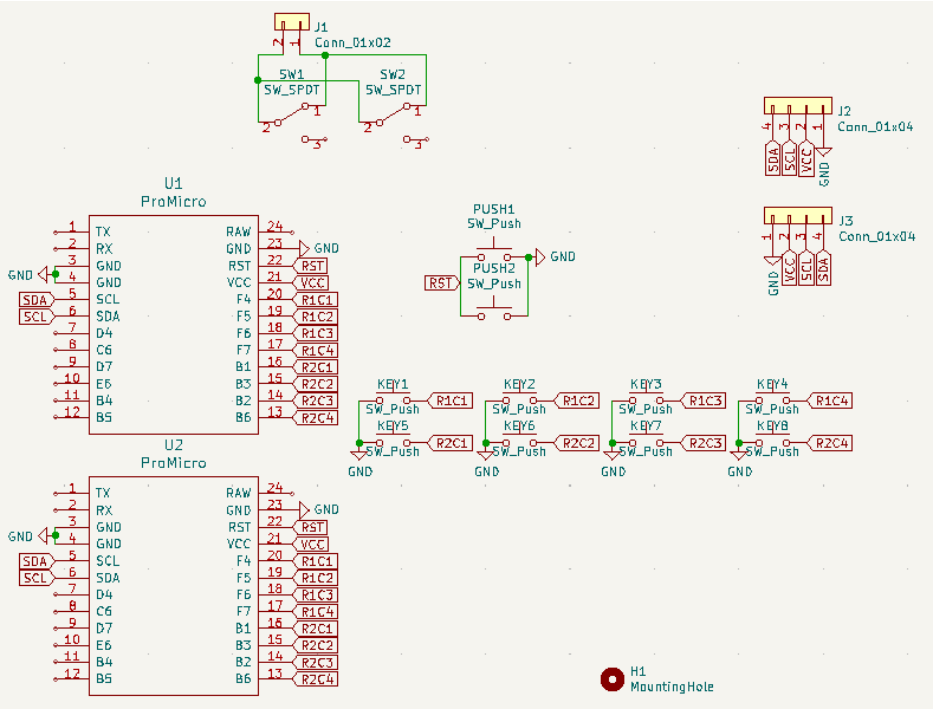

Figure 8.4: HCID v0.3: Electronic wiring schematic

For reference, the average American adult types at roughly 40wpm[5] [52] so we are approaching the range of average performance on a QWERTY keyboard. This is good for a first prototype, however we want it to be considerably faster (approaching 100wpm) for it be truly viable as a keyboard replacement. This may require a revised keymap in the next prototype. Note that there are may features in the pipeline that should help increase the input speed such as whole word input and autocompletion. Additionally, one notable factor currently impeding wpm is Rocking (§ 8.2.2), the device was noticeably faster (roughly 37wpm) when set down on a flat surface, though this could also be due to the drawbacks of our ergonomics.

Note that standard typing speed tests (such as the ones we used) do not include extensive symbolic (or even numeric) input, testing only on prose (*alphas*). This can hide a weakness of (QWERTY) keyboards: being relatively slow to input these keys. We think non-*alpha* inputs are important and should be included in future testing.

8.2.2 Drawbacks

Slow HCID v0.3 is still too slow, not quite reaching the baseline of standard QWERTY keyboard speeds. There are many things to be done to improve this such as reducing rocking, improving general ergonomics, updating the keymap and adding whole-word input and autocompletion.

Rocking The device is not perfectly secured in the hand resulting in instability/rocking when pressing buttons. This makes it hard to type fast and the ergonomics feel tacky.

[5] Programmers tend to type faster; an average of 55wpm.

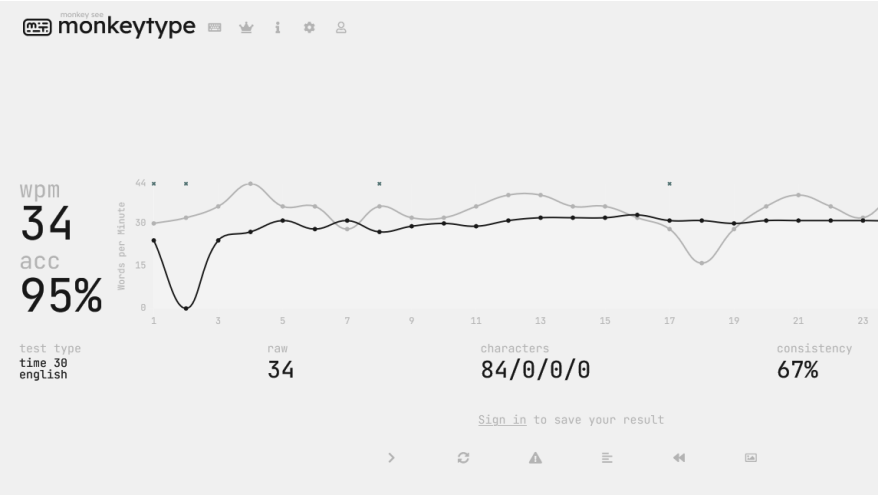

Figure 8.5: WPM progress over time (data collected with Monkeytype [25])

Note how this isn't a problem with (VR) controllers as they don't have buttons for the minor fingers which are free to hold the device. We think this could be mostly overcome by making improvements to the (currently quite rudimentary) strap; to be larger, more supportive and less elastic.

Fatigue and RSI The device is prone to fatigue and RSI. One reason for this is the thin strap provides poor support– pressing into the back of the hand. Another reason is that our choice of key switch has an actuation force considered too high for a chorded keyboard. On a chorded keyboard, you want key switches with very low actuation force because continually pressing multiple keys at once is considerably more strenuous than pressing single keys. Finally, there are still improvements to be made to the ergonomics that would improve the position of the buttons relative to the fingers and reduce strain.

8.2.3 Design goals

Evaluating our Design Goals § 5, we give each a score out of 5 and provide analysis where appropriate.

1. **Hand Controlled** 5/5.
2. **Pervasive** 5/5.
3. **Functional** 2.5/5. Keyboard symbols and mouse controls are complete. This is the minimum viable functionality, however, there is still much more that can be added, including new hardware buttons to support this.
4. **Ergonomic** 3.5/5. Comfortable to hold in the hand and the mechanical key switches feel great. However, the hand strap is not secure enough to prevent rocking and extended use does cause some strain suggesting the button layout still has work to be done.

(a) **Tactile (touch typing)** 5/5. Ergonomic mechanical key switches and having a few keys per finger (two) makes the design highly tactile– you always know what button is pressed without needing to look.

5. **Fast** 2.5/5. Typing speed is currently 30wpm, approaching the average QWERTY typing speed of 40wpm. 100wpm is the next major goal which may require a revised keymap and additional complex features including autocomplete and whole word input.

6. **Spatial** 2.5/5. You can wave your arms around, which is progress, but it doesn't control anything directly (yet).

7. **Isomorphic** 1/5. The design has an isomorphism to letter shapes, but not to QWERTY which is the priority, and could be the key to mainstream addoption.

8. **Extensible** 4/5. Nothing to our knowledge stands in the way of future growth and everything is open source. Perhaps in future, zmk [55] may need to be replaced with custom firmware and our hardware upgraded to include motion sensors for gesture control.

8.3 Conclusions

Overall HCID v0.3 is a usable proof of concept that works well for the most essential set of use cases (keyboard and mouse), it is merely a little slow and rough around the edges. Whilst there is still a long way to go, HCID v0.3 shows that a fast, portable, ergonomic input device is realstic, and provides strong indicators for how we can achieve our remaining Design Goals (§ 5). See Next Prototype: HCID v0.4; § 9) for more.

Chapter 9

Next Prototype: HCID v0.4

HCID v0.3 was the first prototype for which we are confident we got most of the fundamentals right, hence HCID v0.4 will be an evolution on HCID v0.3 (rather than a complete redesign as with previous iterations) that should look and function similarly, with a just few major design decisions to make, some things to directly fix, many things to iterate on, and some new features to add.

Specifically, the sole unsettled major design decision moving forward is for the back buttons; whether to stick with our 4×2 array or switch to a 4×3 (maybe even 5×3) layout, either by squeezing in extra buttons or finding a way to implement our envisioned FBD mechanics. This will be decisive for the keymap design. Easy things to fix include the device rocking, thin strap and high keyswitch actuation. Things to iterate on include the ergonomics, button layout, keymap and pointing-device choice. New features include a custom PCB, comfy strap and perhaps some additional buttons.

Chapter 10

Conclusions

In summary we are pleased with the theoretical and practical progress made towards our goal.

One lesson learned was that we have become increasingly cognisant of the trade-off between theoretical and practical work– better specifying the theory is less time to work on practical engineering.

We would have liked to included far more case studies, perhaps a less detailed but more inclusive table would have been more appropriate.

When we first undertook the project, we had doubts whether it was realistic to build this device at all– especially considering the limited time, resources and expertise. We are aware that this project was ambitious and remains incomplete.

One trade-off we began to view more positively as time went on was 'taking shortcuts'; building on other people's work as much as possible rather than trying to build everything from first principles. This allowed us to focus more on novel parts of the design such as the casing and keymap and less on more 'solved' problems such as key switches and low-level firmware.

Over the course of this project we have become increasing convinced in our central proposition, that the XR revolution is forthcoming, and a fast and pervasive HCID will be required. We hope to be able to continue this work in future.

Bibliography

[1] *(27) CTRL-labs: About — LinkedIn*. URL: https://www.linkedin.com/company/ctrl-labs/about/ (visited on 05/21/2022).

[2] *128X32 OLED Screen*. Little Keyboards. URL: https://www.littlekeyboards.com/products/oled-screen (visited on 05/20/2022).

[3] *5-Way Navigation Switch*. The Pi Hut. URL: https://thepihut.com/products/thru-hole-5-way-navigation-switch (visited on 04/01/2022).

[4] *Adafruit Feather nRF52840 Sense*. Adafruit Learning System. URL: https://learn.adafruit.com/adafruit-feather-sense/overview (visited on 04/04/2022).

[5] *Arduino - Home*. URL: https://www.arduino.cc/ (visited on 04/27/2022).

[6] *ARTSEY — An Easy, Fast, Open One-Handed Keyboard System*. URL: https://artsey.io/ (visited on 04/05/2022).

[7] *Chordite: Yet Another One-hand Keyboard*. URL: http://www.chordite.com/ (visited on 08/09/2022).

[8] Andy Clark and David Chalmers. *The Extended Mind*. URL: https://www.nyu.edu/gsas/dept/philo/courses/concepts/clark.html (visited on 11/02/2021).

[9] Pierre Constantineau. *BlueMicro Firmware*. Apr. 3, 2022.

[10] Ulysse Côté-Allard et al. "A Low-Cost, Wireless, 3-D-Printed Custom Armband for sEMG Hand Gesture Recognition". In: *Sensors* 19.12 (12 Jan. 2019), p. 2811. ISSN: 1424-8220. DOI: 10.3390/s19122811.

[11] *DALL·E 2*. OpenAI. URL: https://openai.com/dall-e-2/ (visited on 08/08/2022).

[12] David Deutsch. "Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer". In: *Proceedings of the Royal Society of London*. 1985.

[13] Doug Engelbart. *Augmenting Human Intellect: A Conceptual Framework*. 1962. URL: https://www.dougengelbart.org/content/view/138/#1a5 (visited on 03/27/2022).

[14] *ErgoMechBoards*. URL: https://www.reddit.com/r/ErgoMechKeyboards/ (visited on 06/10/2022).

[15] Adafruit Industries. *Lithium Ion Polymer Battery - 3.7v 1200mAh*. URL: https://www.adafruit.com/product/258 (visited on 04/10/2022).

[16] *INMO AIR: World's Lightest True AR Glasses*. Indiegogo. URL: https://www.indiegogo.com/projects/2688484 (visited on 08/08/2022).

[17] *Inventor Software — Get Prices & Buy Official Inventor 2023*. URL: https://www.autodesk.co.uk/products/inventor/overview (visited on 04/10/2022).
[18] joric. *Trackpoint Jorne*. URL: https://github-wiki-see.page/m/joric/jorne/wiki/Trackpoint (visited on 12/29/2021).
[19] Nikolai Kardashev. "Transmission of Information by Extraterrestrial Civilisations". In: *Soviet Astronomy -AJ* 9.2 (1964), pp. 217–221.
[20] Aleksandra Kawala-Sterniuk et al. "Summary of over Fifty Years with Brain-Computer Interfaces—A Review". In: *Brain Sciences* 11.1 (Jan. 3, 2021), p. 43. ISSN: 2076-3425. DOI: 10.3390/brainsci11010043. pmid: 33401571.
[21] Edward Laskowski. *Sitting Risks: How Harmful Is Too Much Sitting?* Mayo Clinic. URL: https://www.mayoclinic.org/healthy-lifestyle/adult-health/expert-answers/sitting/faq-20058005 (visited on 07/26/2022).
[22] Wanyu Liu and Telecom ParisTech. "Information Theory as a Unified Tool for Understanding and Designing Human Computer Interaction". In: (), p. 250.
[23] Steve McGowan. "Universal Serial Bus HID Usage Tables". In: (2004), p. 168.
[24] *Mechboards*. Mechboards. URL: https://mechboards.co.uk/ (visited on 04/10/2022).
[25] Miodec. *Monkeytype*. URL: https://monkeytype.com/ (visited on 05/19/2022).
[26] Elon Musk. *Neuralink*. Neuralink. URL: https://neuralink.com/ (visited on 07/12/2022).
[27] *Myo Gesture Control Armband - Black*. URL: https://www.robotshop.com/uk/myo-gesture-control-armband-black.html (visited on 05/21/2022).
[28] Mohammad Namayandegi. "EVALUATION METHOD WHICH PROMOTE CREATIVITY: CASE STUDY ABOUT ERGONOMIC DESIGN IN POINTING DEVICES". In: July 25, 2015.
[29] *Nice!Nano - Nice Keyboards*. URL: https://nicekeyboards.com/nice-nano/ (visited on 04/04/2022).
[30] Peter Norvig. *English Letter Frequency Counts: Mayzner Revisited or ETAOIN SRHLDCU*. URL: http://norvig.com/mayzner.html (visited on 12/28/2021).
[31] *Nreal Air*. URL: https://www.nreal.ai/air/ (visited on 08/08/2022).
[32] *Our World in Data*. Our World in Data. URL: https://ourworldindata.org (visited on 06/02/2022).
[33] Joost Plattel. *Wandering.Computer*. URL: https://wandering.computer/ (visited on 08/02/2022).
[34] *Precision Gaze Mouse*. URL: https://precisiongazemouse.org/ (visited on 08/01/2022).
[35] *QMK Firmware*. QMK Firmware. URL: https://qmk.fm/ (visited on 04/04/2022).
[36] Qualcomm. *XR User Interfaces and Perception Technologies - Qualcomm Developer Network*. URL: https://developer.qualcomm.com/blog/xr-user-interfaces-and-perception-technologies (visited on 08/02/2022).
[37] *R/MechanicalKeyboards for All the Click and None of the Clack!* URL: https://www.reddit.com/r/MechanicalKeyboards/ (visited on 06/09/2022).
[38] Nordic Semionductor. *nRF52840 - Nordic Semiconductor*. 2021. URL: https://www.nordicsemi.com/Products/nRF52840 (visited on 04/04/2022).

[39] Nathan Sharp. *Nazzacode/HCID: My Informatics Final Year Project*. GitHub. URL: https://github.com/nazzacode/HCID (visited on 04/10/2022).
[40] *SimulaVR*. 2021. URL: https://simulavr.com/ (visited on 03/31/2022).
[41] *Steam Deck*. Mar. 22, 2022. URL: https://store.steampowered.com/news/group/39049601/view/5320445713307280739 (visited on 04/02/2022).
[42] *Thru-Hole 5-Way Navigation Switch*. The Pi Hut. URL: https://thepihut.com/products/thru-hole-5-way-navigation-switch (visited on 03/31/2022).
[43] *Tobii Gaming — The Next Generation of Head Tracking and Eye Tracking*. Tobii Gaming. URL: https://gaming.tobii.com/ (visited on 08/01/2022).
[44] *Trackball Breakout - Pimoroni*. URL: https://shop.pimoroni.com/products/trackball-breakout (visited on 04/10/2022).
[45] *Twiddler 3 by Tek Gear*. URL: https://twiddler.tekgear.com/ (visited on 08/09/2022).
[46] *Typeware*. URL: https://typeware.nl/ (visited on 08/02/2022).
[47] Bret Victor. *A Brief Rant on the Future of Interaction Design*. URL: http://worrydream.com/ABriefRantOnTheFutureOfInteractionDesign/ (visited on 06/01/2022).
[48] *VIVE Pro Eye — VIVE Enterprise*. URL: https://business.vive.com/us/product/vive-pro-eye/ (visited on 08/01/2022).
[49] *Vuzix Blade Upgraded Smart Glasses*. Vuzix. URL: https://www.vuzix.com/products/vuzix-blade-smart-glasses-upgraded (visited on 08/08/2022).
[50] *Welcome Home : Vim Online*. URL: https://www.vim.org/ (visited on 04/05/2022).
[51] *Welcome to Meta — Meta*. URL: https://about.facebook.com/meta/ (visited on 05/08/2022).
[52] *What Is the Average Typing Speed, Average Words per Minute?* Online typing. URL: https://onlinetyping.org/blog/average-typing-speed.php (visited on 04/14/2022).
[53] Stephen Wolfram. *Seeking the Productive Life: Some Details of My Personal Infrastructure—Stephen Wolfram Writings*. URL: https://writings.stephenwolfram.com/2019/02/seeking-the-productive-life-some-details-of-my-personal-infrastructure/ (visited on 03/20/2022).
[54] *Zergo Freedom Ergonomics*. Zergotech. URL: https://www.zergotech.com/pages/freedom-ergonomics (visited on 06/09/2022).
[55] *ZMK Firmware*. URL: https://zmk.dev/ (visited on 04/04/2022).

Part IV

Appendices

Appendix A

Keymap Nomenclature

A.1 General

Keymap The mapping of buttons to functionality on a keyboard.

Key Codes Numeric values that correspond to symbols on the keyboard. As a generalisation, any symbol you can think of (`A`, `b`, `VolumeUp`) has a key code. The full list of key codes is given in the USB HID Usage Tables [23].

Layer Maps buttons to key codes. Switching layers remaps all the keys (or as many as you want) to new key codes. Generally a layer is themed to a set of keys, say *numbers*, or *functions*. Layers can be reached either through dedicated buttons or some mechanisms that cycles-through layers.

A.2 Key presses

Where there are multiple names for a behaviour, we have followed the ZMK [55] documentation and alternate name has been shown in brackets.

Send on press Key code sends when you press the key. Standard keyboards work like this.

Send on release Key code send on release of key. Chorded keyboards work like this because humans are not capable of pressing multiple keys within the electronic read cycle of a key.

Toggle Enables some functionality until manually disabled.

Chord (Combo) When a key code is specified by simultaneous press of multiple keys at once. Chords are a way to expand the number of keypress' available on a (small) keyboard; you go from n keys, to $\sum_{i \in n} \binom{n}{i}$ chords (sum of all combinations).

Hold-Tap Will output the 'tap' behaviour if pressed quickly. Will output 'hold' behaviour if held for a while.

Mod-Tap If key is tapped, send key code 'tap'. If key is held, send key code 'mod'. Useful for *modifiers*.

Layer-Tap If key is tapped, send key code 'tap'. If key is held, enable 'layer'.

Momentary layer Press enables a layers while a certain key is pressed

Tap-Dace (Double-Tap+) Enables different behaviour depending on how many times the key is pressed. A common use is `SHIFT` if pressed once and `CapsLock` if pressed twice.

Macro meta-programming functionality to chain other behaviours together with logic. For example, can be used to send multiple key codes at once (e.g. whole words).

Arpeggio (Sticky Key, One-Shot-Mod, Leader Key) enables the user to enter key combinations by pressing keys in sequence rather than simultaneously. Often used for 'sticky `Shift`'– which allows for the writing of a capital without holding and 'sticky layers' which allows the pressing of a single entry from a layer.

A.3 Sets of keys

Commonly referred to groups of keys found on a keyboard. Size of set is indicated in (). To avoid confusion with the usual meaning, set names are referred to in *italics* elsewhere.

***alphas* (26)** Roman alphabet (e.g., `a`, `b`, `c`, etc.).

***numbers* (10)** Numbers `0` - `9`.

***essentials* (4)** `Space`, `Return`, `Tab` and `Escape`.

***symbols* (40)** The standard set of symbolic keys found on a keybaord (e.g., `~`, `!`, `@`, `#`, `$`, `%`, etc.).

***modifiers* (4+)** used in conjunction with another key to modify its normal action. The most important ones are `Ctrl`, `Alt`, `Shift` and `GUI`[a].

***navigation* (10)** Keys for moving around (arrow keys, `Home`, `End`, `PageUp`, `PageDown`, `Delete`, and `Insert`).

***functions* (12)** the F keys (`F1` - `F12`); used to perform specific tasks.

***mouse* (10+)** Mouse movement and buttons (e.g., move, right-click (`MouseButton1`), left-click (`MouseButton3`), scroll-down (`MouseButton5`), `MouseButton7`, etc.)). Unlike most keyboards these will need to be integrated into out design.

Additionally we have some helpful combined sets,

***alphanumeric* (76)** the keys that output symbols: *alphas* + *numbers* + *symbols*

extra symbols any symbols not found on a traditional keyboard (e.g., © (copyright), ™ (trademark), etc.). Currently have to be inserted through the GUI.

all symbols *symbols* + *extra symbols*

[a] `Win` / `Cmd` / `Meta` / `Super` depending on your operating system. They are generally considered equivalent.

Appendix B

Previous Prototypes

B.1 HCID Wristband Concept

Our original HCID idea was for a gesture control wristband that would model Electromyographic (EMG) signals to use hand movements as input.

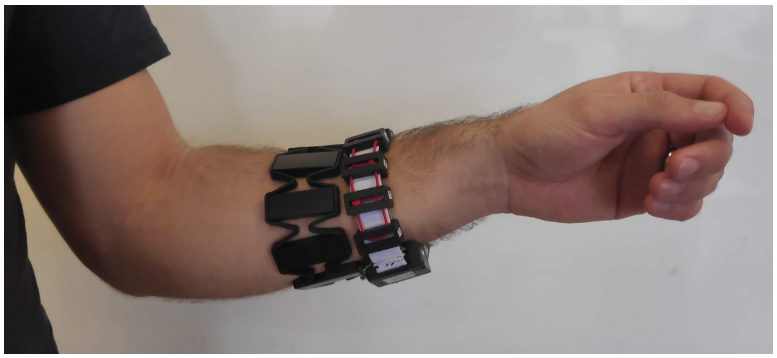

Figure B.1: HCID wristband concept ([10])

We found two companies (and one excellent paper [10]) that had tried this, the first (Thalmic Labs' Myo Armband [27]) was acquired by the second (CTRL-Labs [1]) which was acquired by Facebook, now Meta, who discontinued the public product and integrated into their VR/AR team in 2019 and are still working on the technology. As a result, at the time of writing there are no (longer) commercial products available in this space which is a great shame/opportunity considering what a great idea we think this is. We predict that withing 10 years EMG-wristbands could become humanities predominant input device.

Although we consider an EMG-wristband a more ideal final-form of HCID, ultimately it was considered too challenging a project given the time and resource constraints of the project. For a more detailed discussion see Design Decisions:: Portable keyboard vs. EMG wristband (Section K.1).

B.2 HCID v0.1

HCID v0.1 was the first working prototype. It was designed to maximally simple. It has just 5×1 keys and a very square casing. It used the same keymap as Doug Englebart's original keyset and was implemented on the Arduino firmware platform [5].

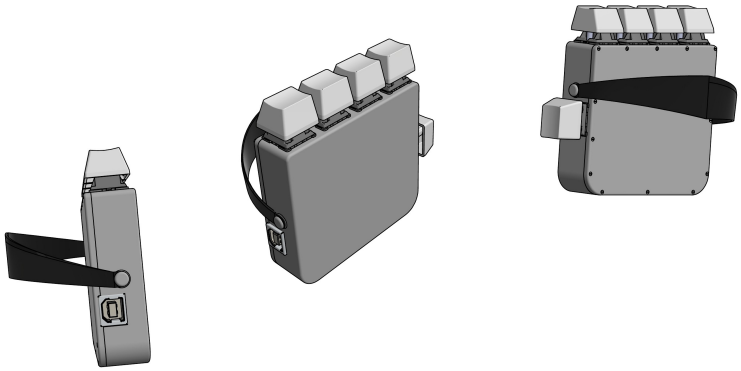

Figure B.2: HCID v0.1: Renders

B.3 HCID v0.2

HCID was a continuation of HCID v0.1, with a joystick replacing the single thumb button and 5-way navigation switches [3] replacing standard (one-way) mechanical switches in an attempt to realise our FBD model (Forward, Backward, Down) (Sec. 4.1). The joystick was a success but the 5-way nav switches were unergonomic and prone to breaking, so swapped back to (two rows of) standard switches for HCID v0.3 (Section 8).

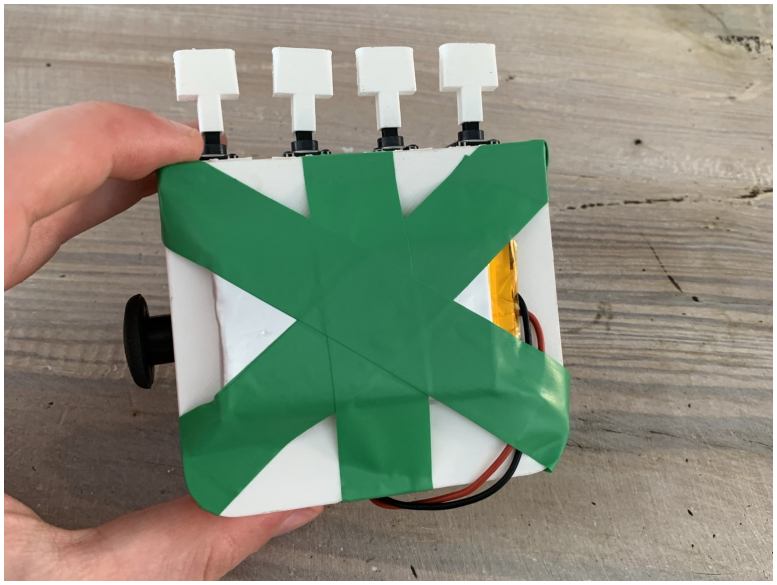

Figure B.3: HCID v0.2

Appendix C

HCID v0.3: Technical Diagrams

C.1 Casing Dimensions

If more detailed dimensions are needed, source files are available in the `/CAD` folder of the main repository [39].

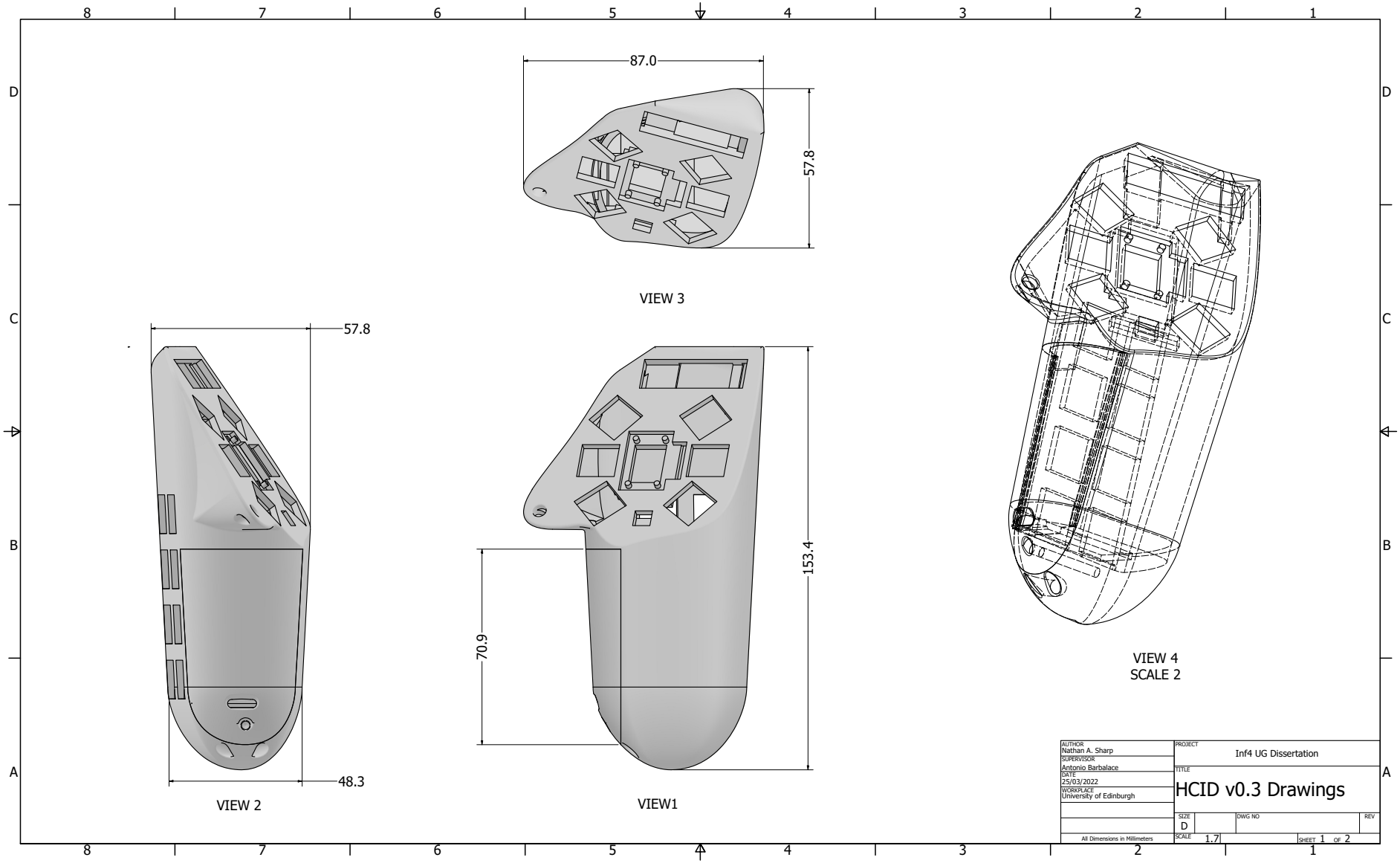

Figure C.1: HCID v0.3: Casing dimensions 1/2

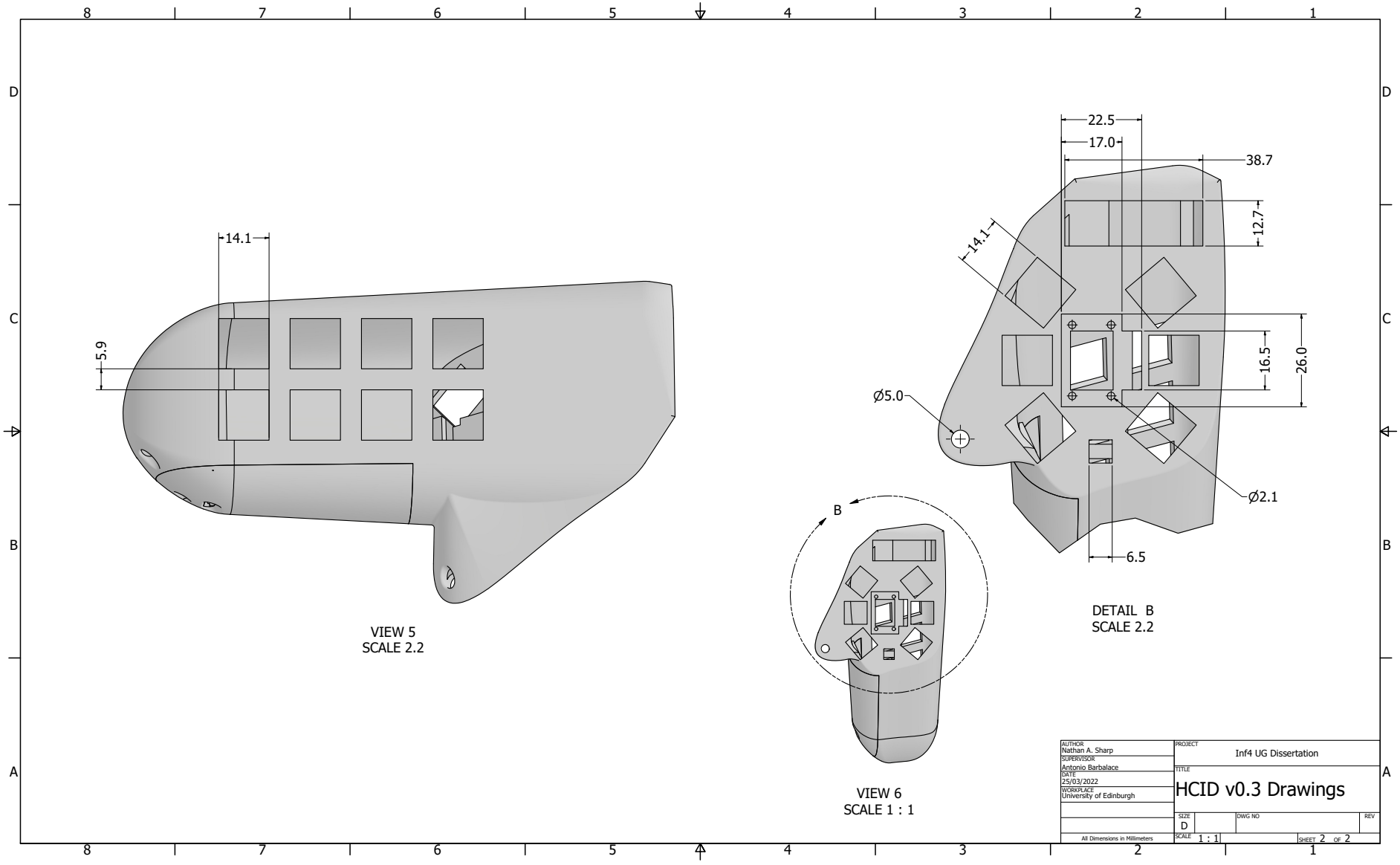

Figure C.2: HCID v0.3: Casing dimensions 2/2

C.2 Keymap Reference

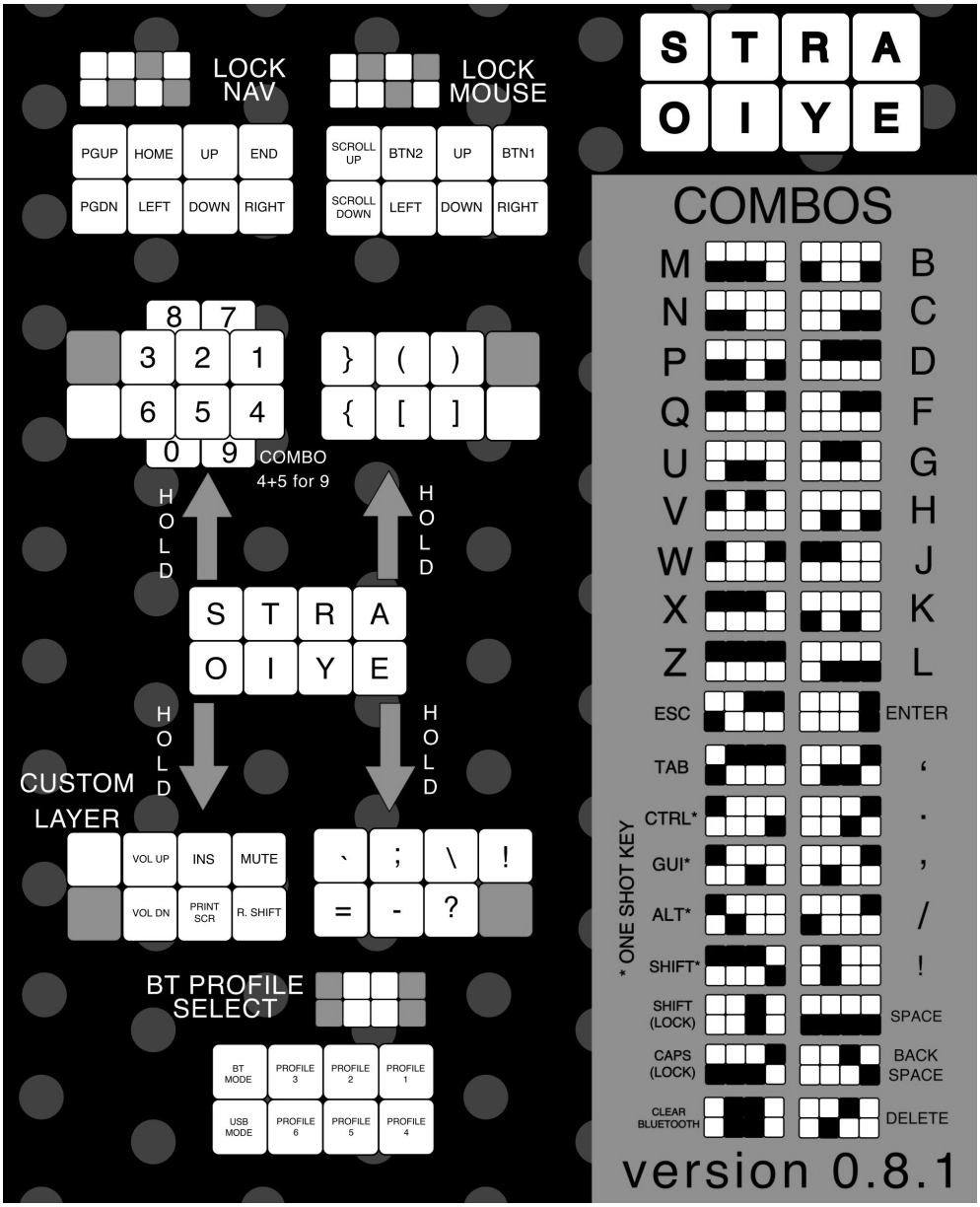

Figure C.3: HCID v0.3: Backside keymap [6]

Appendix D

Brains

Brain (Huamn mind) An individual human biological mind.

Note, we define the term *brain* more broadly than many, and use the terms *brain* and *human (mind)* interchangeably. We (ab)use the term *brain* in this way for no major reason other than to later satisfy the acronym *Brain-Computer Interface (BCI)* with a clearer definition.

D.1 What makes humanity (brains) powerful

Humans are quite simply much more capable than the other animal species we share this planet with. This is because of our brains– we are comparatively more *intelligent*[1]. Intelligence allows us to flexibly manipulate the world to reach our goals.

D.2 What made us increasingly powerful

An obvious source of humanities intelligence is the raw expression of our genes. Yet, despite our roughly constant biological makeup over the past centuries[2], humans have been increasing in *civilisational capacity*– continually compounding our ability to understand and manipulate both ourselves and our environment.

Civilisational Capacity A generalisation of the concept of intelligence to multi-agent collectives[a].

[a] A comparative measure can perhaps be formalised by some quantification inspired by the Kardashev scale [19].

How does this happen? We need concepts that explain this arc of increasing collective

[1] We won't go down the rabbit hole of trying to define intelligence here, but feel confident to accept it as a broad concept of cognitive capacity in which we are generally comparatively more capable than other animals.

[2] And arguably millennia.

intelligence in the absence of genetic evolution. What made human civilisation in 1902 more advanced than in 1290? And likewise in 2019 than in 1902?

Broadly widely cited explanations cover (a) nutrition and literacy, (b) increasing population size, (c) memetic evolution[3], and (d) tool-use/technology.

D.3 What will make us more powerful in future

Presuming that we can continue along this path of increasing civilisational capacity, what will leverage us through an increasingly capable future. For what reasons will humans in 2109 consider 2019 primitive?

Many factors listed in the previous section have reached their limit. Less than 10% of the world is malnourished [32], the global literacy rate for working age people is above 90% [32], and the world population is currently 8.9 Billion– growing slowly and now projected to plateau or even begin shrinking in the next 50 years [32].

From our previously listed factors, only (c) memetic evolution and, (d) tools/technology seem highly relevant to future progress. We also expect new factors driving progress such as (e) artificial intelligence, (f) robotics/automation, and (g) genetic engineering. Interestingly, with the exception of memetic evolution, all factors identified could be considered subtypes of tools/technology.

D.3.1 Tools

We believe the most important factor for much of our past and most of our future progress is that *humans build tools*.

> **Tools (Tool Use)** Artefacts that externalise cognition and action to achieve an objective.

Tools allow us to augment and automate our capabilities with no clear limit, as tools are not constrained directly by humanities physical and cognitive limitations. Note that *technology* is just another word for tools; one with modern connotations.

[3] Fitter ideas and culture survive and propagate. For example societal norms, education and government institutions.

Appendix E

Computers

Computer A device, commonly electronic, that processes information according to a set of instructions.

Over history, humans have created a vast number of tools to empower us further. We've composed wheels, steam, pulleys and gears to pull food from the ground, transport material across continents and construct skyscrapers. The computer was invented less than a single lifetime ago and has already substituted or bettered a growing mass of previous tools; such as the calculator, clock, camera, library and notebook. Furthermore, computers have allowed us to model and directly perceive phenomena that were once beyond our grasp such as the very large, the very small, the very slow, and the very fast. The computer is a very special tool for it broad symbolic-cognitive capabilities.

E.1 Computers make us more intelligent

A computer's abstractions and processing provide a form of leverage, amplifying cognition, thus making humans themselves smarter. This creates a strong feedback loop and compounding opportunity for progress. For example they allow us to capture and calculate the movement of stars with greater accuracy than we could otherwise dream of. In this way, computers amplify intelligence, humanities greatest strength.

Most people only intuit this in a shallow sense, you can caricature this view as 'computers make us more *capable*', downplaying the significance of the situation. The implications of the fuller truth are breathtaking– computers are making us much more intelligent.

E.2 General-Purpose Computing (GPC)

Computers come in many sorts from simple sensors to supercomputers, but for humans, the assumed meaning of the term computer is the *general-purpose (personal) computer (GPC)*.

General-Purpose Computing (GPC) Computing paradigm where the widest possible computing functionality is to provided some individual end user though a single unified interface.

GPC has so far been the dominant interface paradigm defining how humans view themselves in relation to computers. This is in comparison to centralised computers, distributed computers, embedded computers, autonomous computers (AI), etc. Though each of these provides extensive critical functionality at the core of modern society, their output is viewed as secondary to, or in the service of the personal computer interfaces they feed into.

The dominance of personal computers was not always so. In the late 1960s many though that computers would only be used directly by large organisations (centralised compute), exemplified by the sci-fi narrative of computers as large databases that would run planets such as in '2001: A Space Odyssey'. Perhaps one day autonomous AI will become the most important computing paradigm by many metrics. Even so, we suspect that humans would still centre their computing worldview on AI's relationship (interface) with humans.

E.3 Humanity's final tool

Since starting to work with general-purpose computers, we have been in the *computing age*– gradually replacing all other tools with increasingly general computers. Computational theory strongly suggests that a sufficiently advanced computing device could simulate any physical process and hence, **computers[1] are a theoretically complete, fully generalisable tool.**

Church-Turing-Deutsch Prinicple A universal computing device can simulate every physical process[a] [12].

[a] A corollary to this is that the Universe is in some sense a computation. A similar implication holds for the theory of mind, yielding the *computational mind thesis*.

This suggests that all the most advanced tools we will have in the future will be computers[2], making the computer humanities final tool. We don't blink when told that a computer is being used to drive a car, predict the properties of a compound, or even simulate the formation of the Universe. We take it for granted that your standard computer (given enough time and memory) could do just about anything that can be specified. Even retreating to some weaker theory of computability, it is hard to specify anything that one might practically want to do, that couldn't be done with a sufficiently advanced computer.

[1] Equipped with a non-trivial interface to their environment.

[2] And not some *hypercomputer*.

E.4 The progression of computers: Exponential Moore's law

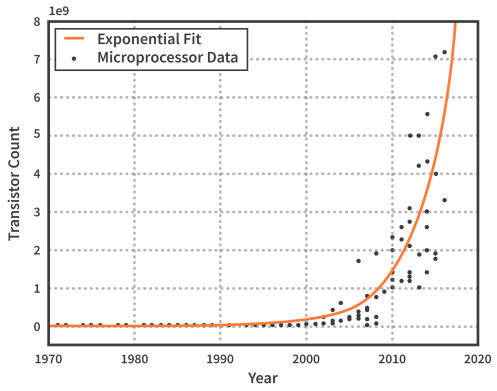

Figure E.1: Moore's law's exponential growth

The rate of progression for computers is staggering, particularly when compared human genetics or (pre-computer) civilisational progress. This is famously illustrated by *Moore's Law*, which observes that **the raw processing power of a computer**[3] **doubles about every two years.** This exponential scaling means that within a few years, the function, size and cost of a computer can change so dramatically as to empower something entirely new. For example, the idea of a personal computer was considered crazy when computers were the size of large rooms and cost more that several family homes. It took visionaries to see how quickly this would change and begin building this future.

E.4.1 Moore's blunder

Exponential scaling (Moore's Law) is notoriously difficult for humans to intuit the effects of, even if they have experienced it already.

> **Moore's Blunder** Failing to recognise how the exponential progress of Moore's Law could soon make a currently smaller/slower/more-expensive computing paradigm viable and competitive for functionality.

Moore's blunder is particularly astonishing when committed by the dominant technological leader of the existing interface paradigm … even more so when their own ascendance owed much to this exact dynamic.

In the 1970's, market leader IBM thought personal computers were a fad, believing they could not have enough power for serious business applications. By the time they were proved wrong with the enormous commercial success of personal computers in 1977[4], it took them four years to release their own in 1981. It was such a rushed and low budget job that they never built –and hence couldn't patent– their own chipset and were soon 'copycatted' out of the market by cheap clones, to the eventual benefit Microsoft who licensed the software on all these devices. Amazingly, despite Moore's blunder enabling Microsoft's ascendance to become the most valuable company in the world, they went on to make an almost indistinguishable miscalculation at the advent of the next hardware revolution– mobile computing (§G.4).

[3] Specifically the number of transistors in a dense integrated circuit.
[4] Specifically the the Apple II, PET 2001 and TRS-80.

In the mid 2000's Microsoft paid little attention to mobile computing[5], as their leadership doubted that a slim pocket sized device could have enough computing power to run serious applications. When they were proved wrong with the enormous commercial success of the iPhone in 2007 (and later Android in 2008), it took them three years to release their own Windows Pho ne in 2010, by which time Android and iOS had entrenched themselves in the 3rd party 'app' ecosystem[6] and Microsoft failed to ever gain mobile market share.

[5]They did develop an somewhat popular PDA (Personal Digital Assistant) platform 'Pocket PC' which crammed a desktop inspired OS into chunky handhelds (see Mobile computing; §G.4)

[6]Not to mention their core OS offering was also relatively poor.

Appendix F

Interfaces

Interface The form of the communication channel between two systems.

We will consider interfaces mainly in the context of *brain-computer interfaces (BCI)*.

Virtually all computers have an interface to a human, from an underwater sensor to VR headset. Abstractly, if information is conveyed in any way, then there is an interface channel through which that information travels. Practically, we reserve the term interface for direct, relatively high bandwidth communication channels. Such interfaces are everywhere. Screens are interfaces, keyboards and mice are interfaces, game controllers are interfaces, sound can be an interface, touchscreens are interfaces, printers are (slow) interfaces, and so on.

F.1 The significance of interfaces

Interfaces define our computers far more than it is generally perceived. They mediate our relationship to a computer so powerfully that they actually strongly define what is possible and what is practical.

Affordance The possibility for action perceived by an agent in an environment. A flat door affords pushing as a computer mouse affords pointing on a 2D plane.

You couldn't watch Netflix on the command line (pre-GUI) and you couldn't send an email out on a walk from a desktop computer (pre-mobile). Additionally, the full utility of smartphones would have been hard to comprehended before they were a reality, simply because it opened up so many possibilities. **A new interface → brings a new interaction paradigm → *affords* revolutionary new possibilities; and there is every reason to believe this will happen again** (see BCI Interface Revolutions; § G). **A core thesis of this work is that interface revolutions are a generally overlooked factor in the story of computing and a presently undervalued target for innovation.**

F.1.1 Interface relativity

Interface devices are so powerful (when combined with computers), that they can literally alter our cognition.

> **Linguistic Relativity (Sapir-Worf hypothesis)** The structure of a language affects its speakers' worldview or cognition, and thus peoples perceptions and cognitive capabilities are relative to their spoken language.

Linguistic relativity is controversial in the field of linguistics as there is considerable debate over its effect size. Holding comment on human-human linguistics, we assert that it applies in it's strongest form for human-computer interfaces, yielding *Interface Relativity*.

> **Interface Relativity** The structure of an interface (to a computer) affects its users worldview or cognition, and thus peoples perceptions and cognitive capabilities are relative to their computer interface.

Consider the cliche example of linguistic relativity– whether color perception varies between speakers of languages that classify colors differently. With interface relativity, there is no denying that you see colors meaningfully differently when looking through thermal imaging goggles: you are literally perceiving parts of the electromagnetic spectrum that are undetectable otherwise. Similar arguments can be made for thinking with a keyboard, python, the internet, a heads-up-displays, etc. Computer interfaces widen the umwelt of available information and cognition in a multitude of directions.

F.2 Information theory for interfaces

> **Information Theory** The study of concepts, parameters and rules governing the transmission of messages through communication systems.

For understanding BCI interfaces, we naturally take great interest in the formal treatment of communication that is information theory. Idealistically information theory could transmute the problem of designing an interface to the problem of designing a strong communication protocol. The fully rigorous specification of BCI protocol is clearly out of reach for the messy, complex communication channels of the real world: cognitive representation, human biomechanics, semantic models[1], etc. We restrict our study to a few simple but powerful concepts that we can interpret into the design of BCI interfaces. Many of these concepts find direct application our 5AA framework (§2).

A note on *words per minute (WPM)*. As you will see, we use familiar concept of WPM as a crude stand-in for *bitrate*. Consequentially, WPM denominates our later approximations of *throughput* for interface devices. WPM breaks down when reasoning

[1] One specific difficulty of rigorously applying information theory to the analysis of symbolic (language) interfaces this there is no concept of semantic meaning in information theory, information is entirely described as a probability distribution so meaning must be crudely approximated.

about input that is not natively denominated in words, such as non-standard symbols, cursor control, and gestures. This can hide the *bottleneck* of an interface, which may have high WPM but be poor at processing these other forms of input. For example, a programmer may achieve 80wpm on a standard (prose) typing test but only be able to achieve less than half that on symbol heavy code. For more on the information-theoretic flaws of WPM and a discussion of alternatives, see [22, pp. 57].

Bitrate The number of bits (information) conveyed per unit time.

- *Interpretation:* The 'Symbol rate' of an interface, how many symbols are input per unit time. Specifically we restrict our quantitative study to WPM (Words Per Minute).

Bandwidth (Peak Bitrate, Maximum Throughput) Maximum rate of information transfer across a channel.

- *Interpretation:* Maximum WPM of an interface.[a]

[a]Note that elsewhere, *bandwidth* is often utilised as the principal variable to maximise. We have opted instead to focus on *throughput*, to (1) capture the true transmission rate– including *noise*, and (2) disambiguate from the 'frequency range' definition of bandwidth used in signal processing.

Bottleneck Minimum rate of information transfer across a channel.

- *Interpretation:* Minimum WPM of an interface.

Throughput The real average rate of successful transfer.

- *Interpretation:* Actual operational WPM.
- *Note:* **This is the principal variable we are trying to maximise in our interface design.**

Data Compression (Coding) the process of encoding information using fewer bits that the original representation. Can be *lossy* or *lossless* depending on whether the original can be fully reconstructed from the compression.

- *Interpretation:* 5AA Abstraction-compression (§2.1.1).
- *Note:* Also consider the adjacent data science concept of

  Data enrichment Processing to enhance data quality by extrapolating missing or incomplete date.

Noise Any additional signal which interferes with information between source

and destination. Often used in the context of *Signal-to-Noise Ratio (SNR)*, a fractional measure of information not related to the message.

- *Interpreted example:* A wrong input keypress[a] or messy sensor reading[b].

[a] Caused by factors such as poor ergonomics.
[b] Such as in eye tracking.

Redundancy The degree to which information is not unique in the system.

- *Interpreted example:* Letter based spelling, a syllable based system has fewer tokens per word.

Latency (Lag) Time delay sending a message from the source to receiver in a system.

Congestion Reduced quality of service that occurs when a network node or link is carrying more data than it can handle.

- *Interpreted example:* Losing your train of thought when an input device (e.g. keyboard) fails to keep up.

System Uptime Proportion of time the communication channel is working and available. Its opposite is downtime.

- *Interpretation:* Accessibility of the interface.

Fitts' Law The time it takes someone to select an object is proportional to 1) how far they are from the object, and 2) the size of the object.[a]

- *Interpretation:* This is most commonly applied to pointing at menus on a GUI, however we think it also can be applied to the button layout of keyboards.

[a] This is a predictive model for the speed of human movement.

Range (Frequency Band) The range of a spectrum over which (satisfactory) communication occurs. This concept allows a system to be decomposed into many discrete channels in which the bandwidth/noise/etc. may be treated differently.

Kolmogorov complexity Length of the shortest computer program[a] that produces the subject as output. It is a measure of the computational resources needed to specify the object.

- *Interpretation:*

  Cog-mogorov Complexity A measure of the cognitive resources required to transmit a message over an interface.

[a] In a predetermined programming language.

F.3 The progression of interfaces: The interface revolution cycle

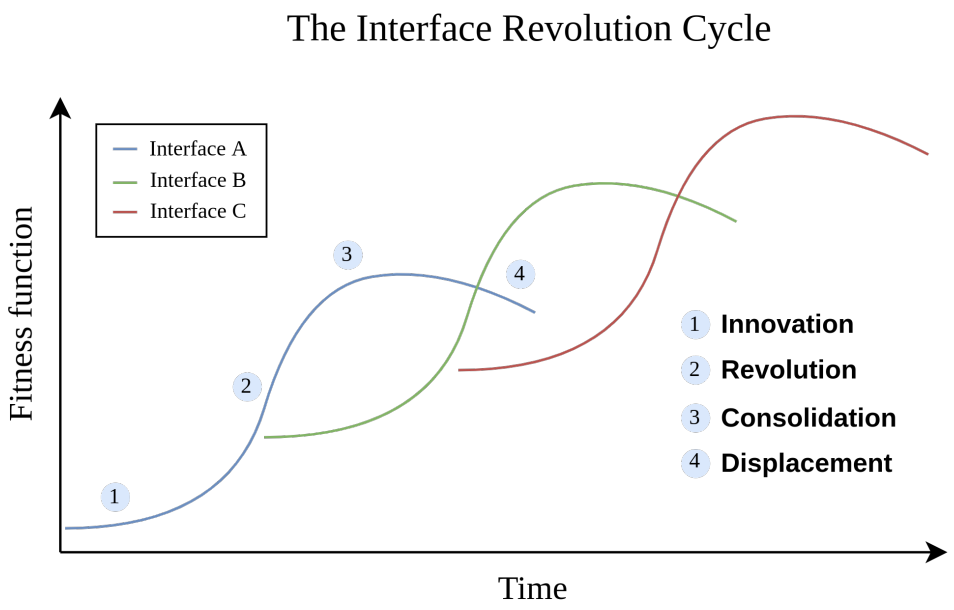

Figure F.1: The interface revolution cycle

Interfaces progress in cycles of 1) an *innovation* period where early adopters experiment and build out functionality, 2) an *interface revolution* where the new paradigm is rapidly adopted and many opportunities are realised, 3) a *consolidation* period of incremental progress where the paradigm matures, and 4) a *displacement* period as the next interface revolution (partially/fully) supplants its functionality. Examples include the adoption of keyboards, personal computers and smartphones. Note that for illustrative purposes the curves in Figure F.1 are uniform, in reality they would exhibit considerable deviation.

We think the study of these (BCI) interface revolutions (§ G) provides a rich framing narrative for understanding the progress of computing systems. For example, when the keyboard revolution (§G.1) replaced punchcards/tape (§G.0) as the dominant input device, it was so dramatic that it 1) required completely new hardware and professional training to operate, 2) redefined software, allowing programs to be written a) digitally, b) targeting a *compiler*– which powerfully allowed for arbitrary abstraction, and (c) in 'high-level' natural language[2]. This resulted in 3) an fabulous widening of access

[2] Specifically a semi-formal subset of English extended with logic notation.

though the already familiar abstraction of the QWERTY keyboard, closing the gulf between the previously separate professions of programming and logic/hardware design. Similarly revolutionary dynamics occurred with every interface revolution (§G).

F.3.1 Causes: switching costs and network effects

What causes this cycle of revolutions? In evolutionary terms, interface design is a fitness landscape where design flows towards a local optima (top of a hill). **Potentially beneficial changes require crossing a fitness 'valley' which incurs *switching costs*[3] and must overcome *network effects*. This leads to interfaces getting 'stuck' at a non-ideal local optima for a long time before the switching costs/network effects are overcome by a considerably better alternative in a 'revolution'.**

> **Switching Costs** Costs incurred by a user as a result of switching products. Switching costs can be financial, psychological, effort-based, time based, etc. The greater the switching costs, the greater the challenge to convince users to switch.

> **Network Effects** Phenomenon where the value of a product depends highly on the number of existing users. It applies when the experience of a product improves as the number of participants increases. Classic examples include the internet and social media. Network effects can entrench incumbent solutions which provide comparatively little value other than their network. New products can be seen as needing to reach a (prohibitively difficult) *critical mass* for the network effect to positively take hold. A famous idiom of network effects is *Metcalfe's Law* which states that the value of a network is proportional to the square of the number of users (n^2).

We find this model to be highly explanatory, for example in indicating why moderate improvements to interface devices (such as non-standard keyboard layouts; §I.2) don't go mainstream– they fail to be better enough to overcome the switching costs and network effects for a critical mass of people. When the next interface revolution breaks through, its can draw on the many minor improvements that were not alone enough to reach mass adoption.

Note the disanalogy to the progress of computers vis-a-vis Moore's Law (§ E.4). Progressing from an intergrated circuit with 100 transistors to one with 1,000,000 was a smoother curve of many complementary innovations, free from singular revolutions that displaced the existing paradigm[4].

[3] For example having computers (and humans) migrated to using keyboards.
[4] Quantum computing could become a notable exception to this rule.

Appendix G

The BCI Interface Revolutions: Past, Present and Future

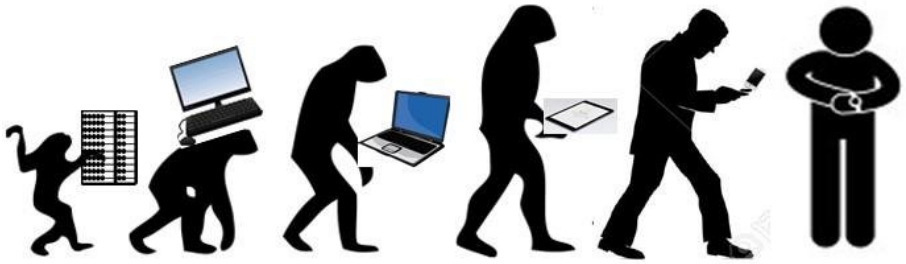

Figure G.1: The BCI interface revolutions

We have already established that whilst raw computer functionality progresses in a approximatly continuous evolution (§E.4) as per Moore's Law, interfaces progress in revolutionary cycles (§F.3) driven by network effects and switching costs. **We now present an opinionated history and futurology of these BCI interface revolutions to help understand and navigate the current moment.** Each revolution upgrades the 5AA interaction model, affording countless new opportunities. **It is hard to overstate the impact of these revolutions; with each we become considerably better entwined with computers, making us more intelligent and unlocking new innovation and social change.**

With each revolution, we observe the following cyclical structure:

1. Visionaries imagine the revolution many years in advance.
2. Subsequent attempts to engineer the revolution are unsuccessful as computers remain too large/slow/expensive[1].
3. unsuccessful attempts lead many to Moore's Blunder (§E.4.1)[2], overlooking the revolution's feasibility and opportunities.

[1] Waiting for Moore's Law to catch up.

[2] Most importantly market leaders from the previous revolution.

4. Progression Moore's law abruptly makes the revolution feasible and the revolution is pioneered by a small group of early innovators.
5. The revolution ascends rapidly, locking in early pioneers with their design decisions and displacing the incumbent interface paradigm(s).
6. The revolution empowers more uses than anyone predicted.
7. Progress stagnates and they cycle begins again.

G.0 Punchcard computing

Operable interfaces for computers evolved gradually. An early milestone was Charles Babbage's design of the Analytical Engine (1837), which (hypothetically) had an input of *punch cards*– stiff paper with a 'program' in patterned holes; an innovation from mechanical looms. Note, 5AA mental impedance is high as humans do not naturally think and communicate in punched-holes. Punched holes were still the dominant input method over 100 years later when the state-of-the-art Colossus computer used tape input to crack the German Enigma Machine (1944). Typically this punchcard input was paired with outputs such as a printer or array of lights.

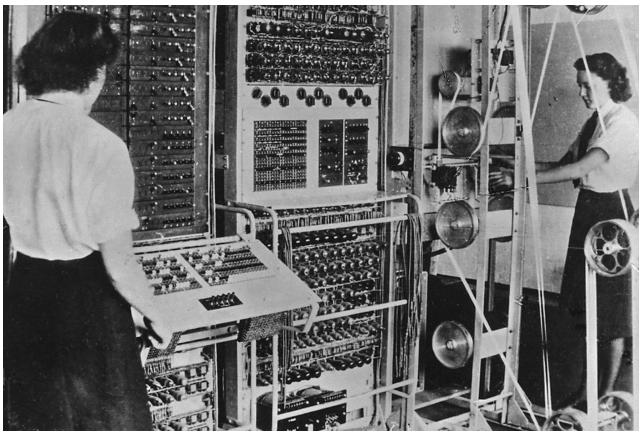

Figure G.2: Colossus code breaking computer (1944) with paper-tape input

In this paradigm, being a direct user of a computer, a *programmer*, was such specialised and complex job that it was at least partially separated from the job of actually conceiving/writing the programs. In 5AA terms, latency, mental impedance and abstraction are all so poor, that operation involves a second human. One breakthrough that helped advance from this interface paradigm was the *compiler* (≈1952), which allowed the input program to be arbitrarily abstract; crucially, mapped into the familiar symbol-set (numbers, letters, etc.) of natural language, making way for the keyboard.

G.1 Keyboard computing

The *keyboard computing revolution* reduced latency to the limit (of feeling instantaneous for humans) and brought the already familiar abstractions of natural language and the typewriter layout. The keyboard computing revolution was principally a revolution in 5AA 1) latency, 2) abstraction, 3) throughput and 4) mental impedance.

By the time electronic input for typewriter keyboards became viable, the typewriter was

already over 200 years old and in large scale use (5AA-familiar). Arguably the first computer with keyboard input was the 'BINAC' (1948) which used a *teletype* system– a halfway-house from punchcard computing where an electro-mechanical typewriter printed data onto magnetic tape which was later used as input. This confirmed the great convenience of keyboard input, which quickly became standard. In 1964, M.I.T and Bell Labs debuted the MULTICS computer, which had direct keyboard input to a *Video Display Terminal (VDT)*– the keyboard as we know it today, direct and instantly responsive. This improvement to latency made communicating commands and controls far more efficient, by the mid 1970's almost all computers used this method. As an idea of how powerful this made computers, almost all the complex internals of modern computer architecture was worked out in this period between the keyboard and the personal computing revolution.

Note that at first keyboard layouts were far more varied than today. With significant influence from the IBM model M keyboard, the modern standard layout crystallised in the 1980's as computers became consumer devices in the personal computing revolution.

G.2 Personal computing

In the *personal computing revolution*, computers became affordable consumer products that could be stationed in the home. The personal computing revolution was principally a revolution in 5AA 1) accessibility, both a) cost, and b) pervasiveness.

The personal computing revolution was arguably the most vibrant of all the revolutions, transforming computers from being specialised business equipment to something everyone used as was essential to everyday life. The landmark breakthrough for personal computing came in 1977 with the release of 'the trinity'– the first three commercially successful personal computers; the Apple II, Commodore PET, and Radio Shack TRS-80, selling millions of units between them. Though individual consumers now had access to abundant computing resources, there was a sense in which it was still the domain of geeks and hobbyists. This all changed with the introduction of the graphic user interface.

G.3 Graphic User Interface (GUI) computing

The *Graphic User Interface (GUI) revolution* provided users with a highly intuitive interaction model, truly bringing computing to the masses. The GUI revolution was principally a revolution in 5AA 1) abstraction, particularly a) familiarity, and related, 2) mental impedance.

GUIs begun with Douglas Engelbart's 'Augmentation of Human Intellect' project at the Stanford Research Institute (SRI) where he developed the idea that computers should be controllable through hand-eye coordination and discovery, in analogy to how children learn, rather than though command languages. He went on to develop the 'oN-Line System' (NLS), a computer which incorporated multiple windows and his specially invented new navigation tool, the computer mouse (§I.3). This work led directly to

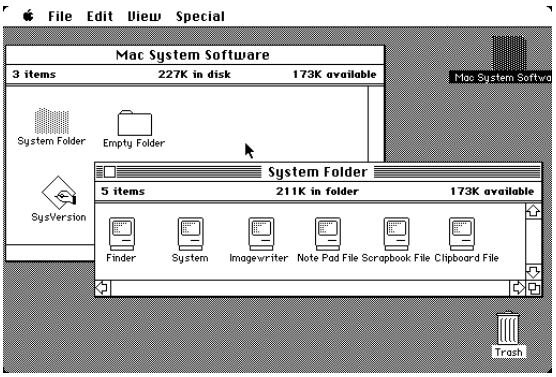

Figure G.3: Apple Macintosh GUI (1984)

the progress at Xerox PARC, where they soon demonstrated a GUI with (the now commonplace) windows, icons and menus (1974). Apple had been shown the mouse and GUI at Xerox PARC (1979) and subsequently scrapped existing plans to redesign everything around this new paradigm. They released the comercially sucessful GUI operated Apple Macintosh in 1984[3] (see Figure G.3), leading the ascendance of the mouse operated GUI as the de-facto computing interface.

G.4 Mobile computing

The mobile computing revolution empowered taking capable and inexpensive personal computers almost anywhere and prompted a further simplification of the UI interaction model. The mobile revolution was principally a revolution in 5AA 1) accessibility– both a) cost, and b) pervasiveness, and 2) mental impedance.

Portable computers were a widely foreseen continuation from the success personal computers as Moore's Law continued to push the bounds of computing. Alan Kay had one of the first practical visions for a portable computer in the 1970's, the 'Dynabook', and although it was never developed, it is credited with inspiring many following attempts. Through the 1980's and 90's a progression of gradually more portable computers evolved into to the modern 'clam shell ' laptop, a pervasiveness middle ground between desktops and what would soon be pocket sized mobile devices[4].

Advancing pervasivness beyond the laptop, the first *PDA* (*Personal Digital Assistant*), 'The Organiser', was released in 1984 by Psion. These were limited functionality handheld computers, operated by a mixture of miniature keyboards, buttons and styluses. Through the mid 1990's many company's released popular PDAs, and in the early 2000's these devices merged into smartphones as they were increasingly equipped with cellular capabilities. A final near-miss before modern smartphones were 'Pocket PCs' (mid 2000's)– devices which had more featureful operating system than previous PDAs, running Windows Mobile OS. In hindsight, they did not comfortably fit in a pocket and were too close to desktop operating systems– cumbersome and relatively complex

[3]The high end, GUI operated Apple Lisa was released a year earlier (1983).

[4]A laptop can be transported to any location, but can't be fit in a pocket or practically used on the move.

to operate. Note that in a classic case of Moore's blunder (§E.4.1), Microsoft paid relatively little attention to their development, their leadership incorrectly reasoned that mobile was an underpowered and fringe computing platform.

The watershed moment for the mobile computing came in 2007 with the release of the Apple iPhone. It debuted essentially all the remaining interface features of modern mobile devices. It had a large multi-touch capacitve touch screen (see Touchscreen Interfaces; §I.5), a simplified mobile operating system (with the now iconic app grid and 'home row') , the best mobile web experience to date, and 1st class support for third party applications ('Apps'). Additionally they dropped the stylus– allowing the device to be used one handed[5], and physical keyboard– an ineffective waste of space (see Touchscreens; §I.5 para. 3).

Note that mobile devices generally have lower throughput than desktop devices due to factors such as limited screen real estate and slower input (touchscreen typing). As such mobile is not preferred over desktops for many of the most serious computing applications. **This is somewhat unique among interface revolutions: Mobile has only partially displaced the desktop interaction model.**

Remark that, whilst mobile revolutionised pervasiveness, mobile devices are far from being fully pervasive. Though they can be transported anywhere, usage has considerable frictions such as 1) startup latency– removing from pocket and unlocking, 2) physical impedance– a) needing at least one hand at all times (for both input and output), and b) unergonomically having input and output fused in the same place (e.g. no heads up display), 3) mental impedance– the need for constant gaze as touchscreen input is not tactile. This secludes mobile from a large range of use cases. For example, simply walking whilst using a mobile device is difficult, let alone anything more physically active.

As of 2022, we are witnessing the tail-end stagnation of mobile, making way for the coming revolution.

G.5 XR computing

XR (Extended Reality) Computing environments that are almost totally immersive[a] (VR) and/or pervasive (AR); tending to provide an overlay/alternative to 3D-spatial and sensory reality.

[a] aka very high output bandwidth.

At the time of writing we are at the onset of *XR computing revolution* (alt. *wearable computing* revolution). In the XR revolution computers become i) *totally immersive*– capable engaging the entire field of perception, and ii) *almost fully pervasive*– available for use in almost any situation. In 5AA terms, this 1) overhauls throughput– massively increasing bandwidth, whilst decreasing noise, 2) pushes pervasiveness near its limit– becoming constantly 'wearable', and (3) considerably reduces impedance, both physical,

[5] Resulting in less physical-impedance and hence greater pervasiveness.

Figure G.4: Artists impression of an XR interface

and mental. **We predict that XR will combine the best qualities of desktop and mobile in a single interface and replace them both.**

Note, our focus is on XR as a GPC platform for maximally augmenting human intelligence. This puts some distance between us and the in-vogue (narrower) outlook of XR as a social-interactive entertainment metaverse of 3D virtual objects[6].

One of the first visions of *VR* occurs in the novel Pygmalion's Spectacles (1935), as goggles that transport you to a simulated holographic world. In 1958, the first VR machine, the Sensora, used stereoscopic lenses to show short films and in the 1960's similar displays became head mounted. The Sword of Damocles (1968) was a landmark VR/AR headset, connected to a computer rather than a camera. VR progressed gradually over the next decades finding use in flight simulators and military headsets. The first mass produced VR arcade machine came in 1991 and the first home VR headsets came in 1995. The late 1990's and 2000's were a period of relative public and investor disinterest in VR. We think VR was impeded from mass market viability by the need for (expensive) high powered CPU's[7]. Such a CPU, which would also be heavy and large would also need to be either 1) head mounted– increasing weigh so reducing comfort, or 2) grounded/worn elsewhere– requiring impractical wiring. In 2010, Palmer Lucky recaptured the public's imagination with his design for the 'Oculus Rift', a modern VR gaming headset, beginning the wave of innovation that we are currently riding (2022). In 2021 the Oculus headsets outsold the Xbox and there are hundreds of companies developing VR products.

AR arguably begun with simple *Heads-Up Displays* (HUDs) that were developed for pilots in the 1950's showing a laserbeam overhead. The history of AR and VR overlapped with the aforementioned 'Sword of Damocles' (1968), and in the following decades, companies, universities and governments further advanced AR technology for wearables and digital displays. Steve Mann coined the term *wearable computing* in 1980

[6]We call this the 'Zuckervision'.

[7]To (1) run high resolution displays (for the large field of view), and (2) do intensive 3D rendering.

and Thomas P Caudell (of Boeing) coined *Augmented Reality* in 1990. The first modern AR system was probably the 'Virtual Fixtures' (1992), a complex robotic system that would overlay of sensory information on a workspace to improve human productivity. Though incremental progress continued to be made, particularly in military applications, AR suffered much the same relative disinterest as VR through the late 1990's and 2000's, likely because computers weren't yet powerful enough. A underrated cultural factor in the reinvigorating the vision of XR was the pop iconic 'Iron Man' superhero film franchise (2008, 2010, 2013), in which the protagonists 'superpower' is a technological suit equipped intelligent XR computing environment. In 2014, Google released their limited access 'Google Glass', a pair of glasses with a simple smartphone-esqe AR overlay to one eye, before axing the project in 2015[8]. Google tentatively restarted the project in 2017 focusing on Enterprise use cases, but we gauge that so far it has received relatively little development attention. In 2016, Microsoft released the 'HoloLens', a competitor visor like headset also targeted at specialised business functionality. As of 2022 Apple are rumoured to have both an AR and VR headset in the works. Additionally there are several smaller companies that have been releasing increasingly impressive consumer AR glasses [31, 16, 49] . The current state of the art is a large, high resolution 2D display overlay but little more. **We expect this to take off rapidly over the next couple of years.**

For more, see The Next Revolution: XR Computing (§H)

G.6 Nervous System Interface (NSI) computing

Nervous System Interface (NSI) BCI that targets direct stimulation/reading of the brain/nervous-system[a]. The distinction is often made between *invasive*– requiring implanted surgery, and *non-invasive* NSI methods.

[a]Reminder: what we call NSI is typically referred to as BCI in much of the literature. See Brain-Computer Interfaces (BCI) (§1) for why we deviated from this.

The NSI revolution is the final BCI revolution, where direct electrochemical reading/stimulation will provide a practically infinite-bandwidth brain-computer interface, with qualia of a likeness to natural cognition and experience. The NSI revolution will principally be a revolution in 5AA 1) bandwidth, 2) abstraction, and 3) impedance.

The history of NSIs starts in 1924 when Hans Berger discovered the electrical waves of the brain and fathered *electroencephalography* (*EEG*)– a method to record an electrogram of the electrical activity on the scalp via electrodes. The modern field was pioneered by UCLA professor Jacques Vidal in the 1970's who coined the term BCI and set out the 'BCI challenge'– to control external objects using EEG signals. In academia, brain signal acquisition and processing has made considerable gradual progress in the decades since, finding itself increasingly in the scientific mainstream. So far work has often been with an emphasis on allowing impaired peoples to (re)gain normative control over their lives. The research literature is now vast and a detailed coverage can be found in [20].

[8]Rumored over the privacy concerns of its front facing camera.

One commercial company in this space we would like to mention is Neuralink [26]: a well funded and high-profile attempt to rapidly advance invasive NSI. They are developing an advanced 'sewing machine-like' tool for implanting very thin electrode threads into the brain. Whilst we welcome their ambition (and NSI research more generally), when we consider Neuralink's stated goal of "increasing human-computer bandwidth"[9], we feel there is a largely overlooked –and hence underfunded– valley of opportunity between current BCI paradigms (desktop and mobile) and NSIs: XR interfaces that radically increase human-computer bandwidth and are achievable with today's technology.

[9]We might also suggest they recognise pervasiveness as a major goal.

Appendix H

The Next Revolution: XR Computing

Let us consider the coming XR revolution in greater detail than discussed earlier (§G.5). Specifically, we detail the advantages and opportunities of XR computing in relation to the currently dominant interface paradigm(s) of desktop and mobile.

Worth mentioning is that that XR is closely related to the intuitive concept of *wearable computing*:

> **Wearable Computing** Computing environments with a pervasiveness comparable to clothes[a] – easy to put on and remove, constantly operational, lightly carried on the person, and externally visible.
>
> [a] i.e pervasiveness greater than mobile computing and less than NSIs.

H.1 Output: 3D FOV HMDs

The output form-factor of *XR Computers* is essentially already known, it being their implementation that remains a challenge. Output will primarily be a 3D *Head Mounted Display* (*HMD*) spanning the entire *Field of View* (*FOV*). **In the near future, at any moment, you will be effortlessly able to bring up a interface that spans your field of view with the power of (at least) a desktop computer**. This can range from being totally immersive (VR) to an overlay of reality (AR). As Moore's Law progresses, it will be possible to implement such displays first with large headsets, then smaller goggles/glasses, and (maybe) finally contact lenses.

Note VR (immersive) output is presently further developed than AR output, there already being polished consumer VR gaming headsets. **For XR to become a truly general-purpose computing platform, we think similarly advanced AR output will be needed. This is because the immersive nature of VR is highly prohibitive of essential functionality such as multi-hour sessions and awareness of ones surroundings.**

H.1.1 3D FOV displays

Output throughput is dramatically increased by the adoption of 3D FOV displays. We will analyse both the 3D and FOV parts in turn.

3D display output is important because humans naturally think in three spatial dimensions, which in turn is because the physical world is 3D. Having computing environments that add an extra (already [BROKEN LINK: familiar]) dimension to their representation –and hence our thinking–, will allow us to process more information in many situations. Examples applications might be CAD, social interaction, knowledge work and mixed reality.

FOV displays appear multiple times larger than the typical laptop screen or monitor[1] taking the amount of information that can be processed by the visual system to the limits of human capability. This greater bandwidth, which fundamentally allows more (and more complex) information to be represented between the computer and human, will make us more intelligent. **You accept that there are many computing applications that are impossible or impractical on mobile for the reason of screen size, would a similar relation not hold between monitors and FOV displays?**

The value proposition of FOV is even greater for mobile use cases, as mobile screens are an order of magnitude smaller than computer monitors. When combined with high-pervasiveness (HMDs), FOV displays will grant us to take (better than) desktop computing anywhere.

H.1.2 Head Mounted Displays (HMDs)

Head Mounted Displays (*HMDs*) massively increase pervasiveness by eliminating the physical impedance of holding a mobile device (currently the most pervasive) when in use. They also relieve us from the physical and mental constraints of *stationary computing*.

HMDs transform the situations in which computers can be carried and used. For example, a HMD could be utilised whilst moving, such as when exercising and could potentially provide access to computing a interface during social interaction that is more subtle and less obstructive than a mobile device.

Combined with ultra-high throughput FOV displays, HMDs allow the computing currently constrained to stationary desktops, to be freed into any environment. Ask yourself why computing should be tied to the stationary desk? This is an accepted norm with profoundly damaging consequences. As well as common aches and pains, research found that those who sat for more than eight hours a day with no physical activity had a risk of dying similar to that posed by obesity and smoking [21]. We also predict cognitive benefits of computing environments that embrace natural posture (see Spatial computing; §H.3).

[1] Arguably FOV can currently be achieved with multiple monitors. Note this is both highly useful but also prohibitively expensive and technical to setup, highly UN-portable, and not well accommodated for in OS interface design.

H.2 Input: Unknown (the Achilles heel of XR)

In our study of the BCI interface revolutions (§G), every time there is a significant change to the output interaction model, this compels a corresponding novel input paradigm/device. For example, the continuous 2D spatial environment of the GUI revolution lead to the creation of the computer mouse pointing device.

From the other side, its stating the obvious to say that incumbent paradigms interface devices simply will not do in XR. The keyboard and mouse both require a desk, something we won't be carrying round with us, and touchscreens require impossible panels as large the output display, not to mention them being inherently 2D.

For reasons related to the *iPadification fallacy* (§J) (the assumption that XR will be only like mobile, not desktop) we are confident people are working on a good 3D pointing device solution for XR (we think eye tracking (§3.3) is best). **However, the converse is also true: despite hundreds of millions of pounds flowing into the development of XR, almost no one is explicitly working on high-throughput (symbolic) input devices for XR, to supercede the keyboard and hence desktop functionality. We think this oversight will hold back XR from its near term potential while it remains unresolved.** *Meanwhile, there is a great opportunity and responsibility[2] in filling this void. This is the fundamental logic that motivated our practical project (§III) to design a novel input device worthy XR and why we claim that (high-throughput) input is presently the *Achilles heel* of XR.

See also Common Misconceptions (why nobody is building this) (§J).

H.2.1 Overcoming the limitations of keyboards/mobile

Many advantages of XR stand in relation to overcoming the flaws and limitations of the incumbent interface paradigm, the desktop (monitor, keyboard (§I.1.1) and mouse (§I.3)) and mobile (§I.5) respectively. In brief summary, keyboards have poor ergonomics (causing RSI), low pervasiveness (not mobile), have only moderate throughput, and suffer from the 3 hands problem (§I.3.2). Mobile touchscreens meanwhile, have low throughput– for both output (screen size) and input (typing speed), are not fully pervasive– needing one hand to operate at all times, and are not tactile (cannot be operated without looking)– increasing mental and physical impedance. XR computing stands to fix all of these issues and more.

As seen in I/O interaction model: The inputs (§H.2), we expect the XR revolution to necessitate a revolutionary new input device/paradigm. Remember (from switching costs and network effects; §F.3.1) that there are many established (non-XR) minor improvements to the current interface paradigm that can cascade into an interface revolution, such as the many innovations from ergonomic keyboards (§I.2) They need the shake-up of a revolution to overcome the stickiness of the entrenched paradigm.

Additionally an new XR interface should solve the *two interface problem*, providing a single unified computing interface rather than have users straddled between desktop and

[2] As what catches on in an interface revolution will likely get entrenched through network effects and switching costs (§F.3.1)

mobile environments. Note this doesn't happen unless we get keyboard level throughput, otherwise people will just stick to their desktops.

H.2.2 Radical new opportunities

We have already argued that the XR revolution will necessitate a radical new input device (§H.2), outlined the flaws to be fix of the current ones (§H.2.1), and hinted that this is what our practical project (§III) will tackle. Briefly we would like to cover to some some the radical opportunities beyond merely fixing the flaws of the current paradigm. Specifically the opportunities of drastically increasing 1) input throughput and 2) pervasiveness.

H.2.2.1 XR throughput

As we have already discussed, and later lay out in detail (§I.1.1.4): the keyboard is only a moderate throughput device (30-120wpm), far from what we think are the theoretical throughput limits of BCI (§1.3) ($\geq$350wpm), and there are already devices such as the stenotype (§I.4) and various chrded keyboards that comfortably allow input throughput over 200wpm. **We hope that the XR revolution will integrate this opportunity for much faster input than the keyboard.**

Drastically increasing input throughput leads to a radically different world. Consider the inverse: if you could flick a switch to make everyone permanently type 80% slower (8wpm on average), what would happen? We would be in a slower, less productive, and less capable world? There is a realistic near future where the average person could input 5× faster[3] (40wpm $\rightarrow$ 200wpm), few seem to intuit our conviction that this could itself have revolutionary effects.

H.2.2.1.1 The true cost of low throughput input

We also think there are some non-obvious hidden costs to low(er) throughput input. We attest that there are both the (more obvious) *quantitative costs* of being slower and wasting time, and the (less obvious) *qualitative costs* of cognitive load congestion– interrupting the stream of consciousness, breaking up ones ability to form the most complex thoughts. An example of the second is when you forget the details of a sentence because you can't copy it down fast enough.

H.2.2.2 XR pervasiveness

High pervasiveness will also becomes an incredibly powerful force multiplier of possibility. As soon as you accept the likelihood of AR HMDs, you want an input device to match this (unprecedentedly high) pervasiveness (with reasonable throughput). This unlocks many aspects of Spatial computing (§H.3) allowing computing configurations such as *strolling desks*.

[3] As measured by WPM and we think this is a justified approximation of input more broadly.

H.3 Spatial computing

Spatial Computing Computing interfaces that leverage our knowledge and familiarity of objects and movement in 3+1 dimensions.

Figure H.1: Superhero Tony Stark (Ironman) interacting with spatial computing XR interface 'Jarvis'

The powerful abstraction of spatial computing has been limited in computing interface paradigms so far (e.g. the mouse could be considered rudimentary 2D spatial computing) and now has a chance to take centre stage with XR. It's value proposition is supported by arguments from the literature that the movement and manipulation of objects in 3D space is central to and even involved in cognition (see embodied cognition in §1.3). **This would imply that there has not yet been a computer interface paradigm native to human movement and spatial thought.**

Many claim[4] to do much of their best thinking whilst walking, counting some of the greatest philosophers (Aristotle, Nietzsche, Kant, etc.) among them. How much human productive capacity has been left on the table by the cementing of computers –and hence human thinking– to stationary desks? Anecdotally, one often becomes physically restless sitting at a desk computer long before suffering mental fatigue. We predict spatial computing could be the wow-factor that makes XR computing 'feel' like a revolution, distinctly separating it from what came before.

[4]Including this author.

Appendix I

Case Studies *or* Doug Engelbart and the Keyset

This section contains our curated selection of cases studies on BCI input devices that were influential to our work, analysed through our 5AA framework (§2).

Interestingly, since the decline of (physical) interface design research in academia (late 1980s onwards), the best sources of knowledge have tended not to come from academia[1], but rather from online sources: start-ups, blogs and builder communities (Reddit, Discord, etc.). The exception to this rule is Nervous System Interfaces (NSIs) which remains almost entirely rooted in academia.

The major theme running through our cases studies is the flaws of the typewriter keyboard and their prospects of salvation.

There were many additional case studies that we did not get to include here, examples include the game controller, the CTRL-Labs Wristband [1], Neuralink NSI [26] and the Google Glass AR glasses.

I.1 Typewriter Keyboard

They modern (typewriter style) physical keyboard is presently one of the most ubiquitous and powerful tools in existence. It is a high-bandwidth symbolic input device, that, together with the mouse **dominates the professional and high performance computing landscape where there are effectively no market share competitors**.

In terms of total computer usage, the keyboard & mouse currently has approximately equal market share to mobile devices by metrics such as 'time spent web surfing', with mobile devices steadily increasing in popularity over the past decade due their ease of use and accessibility (greater pervasiveness and reduced cost). For example, most people in developing countries only have a mobile device.

[1] We theorise this change occurred because (1) building experimental interface devices became considered less academically "pure", and (2) an unspoken assumption developed that the keyboard & mouse was here to stay.

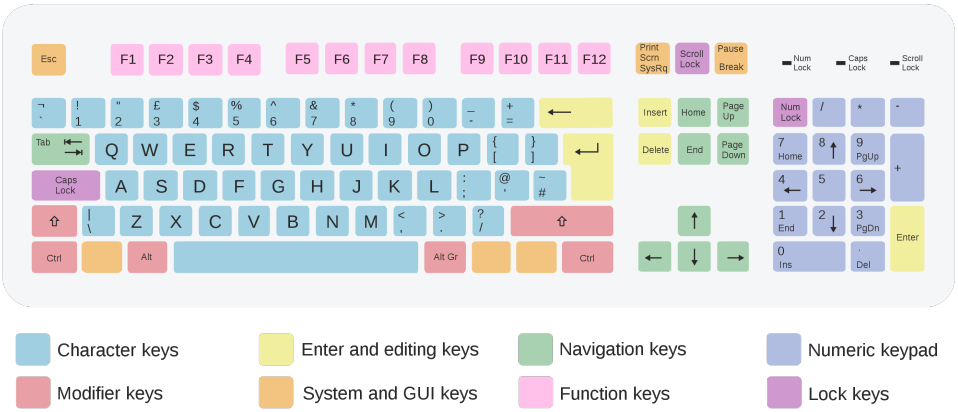

Figure I.1: Standard 105-key qwerty keyboard

Tablets, notably, are increasingly appropriate for many computing use cases where a keyboard & mouse would have otherwise been used, but, as a rule, this tends to be for the more 'casual' use cases. For the most productive work, the relative higher throughput (WPM) of a physical keyboard with tactile keys is strongly favoured which means that **no touchscreen is every likely to be preferred over a keyboard for high productivity input**. As mobile computing also has situational necessity, we are currently stuck with the *two interface problem*.

The typewriter keyboard in a visually recognisable form can trace its roots back as least as far as 1710. The standard Qwerty layout was invented for typewriters in the 1870s[2]. The keyboard only became the de facto symbolic input device for computers in the late 1950s, previously computers were predominantly controlled using punchcards and derivatives such as tape (§ G.0). The modern standard layout (Figure I.1) crystallised in the 1980's as computers became popular consumer devices.

From its 18th century origins, **the design has changed surprisingly little despite the fact that many decisions were made for contingent reasons that are now redundant.** For example, mechanical typewriters were designed around physical levers, favouring a 2D-grid tabletop design, a constraint that does not applies to modern electronics. Meanwhile, evolving technology has created new use cases (§6) that should be incorporated into the design goals (§ 5) of a modern symbolic input device.

I.1.1 The problems

We will now consider some of the flaws of the keyboard in detail, specifically with respect to our 5AA interface framework (§ 2) and our vision of the (XR) future (§ G.5).

Note, many of the discussed problems have already been fully or partially solved, sometimes in several different ways, most commonly by ergonomic/custom keyboards (§ I.2). We have tried to make note of when this is the case.

[2]And became the standard layout from the 1920s onwards.

I.1.1.1 Poor ergonomics

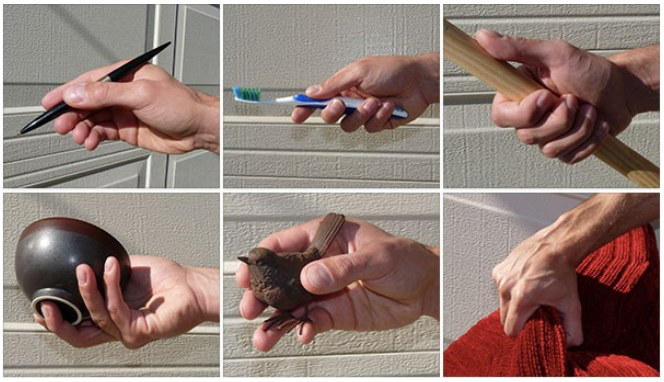

Figure I.2: Some natural hand grips ([47])

Keyboards seem obviously unergonomic with respect to the hands that operate them as the 2D-grid layout does not induce the natural posture of the hand (see Figure I.2) Note that generally ergonomics fall under our 5AA attribute of physical impedance (§2.4.1), though there are many other factors at play. Specifically this unergonomic posture excludes any opportunity to leverage familiarity (§2.1.2) (see also Spatial computing; §H.3) and chronically induces strain and injury (see Repetitive Strain Injury (RSI); §I.1.1.1.1). Furthermore, **although better than most alternatives, keyboards are not fully tactile**– they cannot always be easily operated without looking; many of the buttons are in difficult to reach locations away from the 'home row' (consider how this is not the case of a game controller)[3], so unsurprisingly, the majority of people never learn to fully touch type. Another weakness of standard keyboard ergonomics is the underutilisation of the thumbs, they are the strongest and most manoeuvrable finger, yet the standard keyboard layout has both thumbs controlling just a single shared button, the space bar[4].

Figure I.3: Strained wrist position on a standard keyboard ([54])

[3]Solved by custom keyboards using chords/layers.

[4]Again, solved in many custom keyboards with 'thumb clusters'.

I.1.1.1.1 Repetitive Strain Injury (RSI) **As a consequence of poor ergonomics, recurring back problems, sore forearms and (most commonly) wrist pain are all endemic amongst regular keyboard users.** These are all forms of *Repetitive Strain Injury (RSI)*, which occurs when tendon tissue slowly degenerates as a result of constant stress or strain. One study suggested that 60% of those who professionally use a keyboard will suffer from RSI at some point [28]. **We suspect there is no human tool more commonplace associated with such unnecessary widespread pain and injury.** This situation is not unpreventable as effective solutions do exist in the form of ergonomic keyboards (§I.2).

I.1.1.1.2 Qwerty The standard 'Qwerty' keyboard layout was originally designed to prevent permanent print errors on paper[5], spacing associated letters as far apart as possible. **This redundant centrepiece of the qwerty design is antithetical to both reducing strain and increasing speed as it maximises the distance fingers have to travel.**

The Qwerty layout has many futher ergonomic faults. Firstly, the offset between rows are diagonally staggered[6] which causes extra strain even compared to an ortholinear (chocolate bar) grid where fingers move in more natural straight lines. **Additionally, the layout does not make it easy to reach the most common keys– generally considered the most important priority in keymap design. For example, in English, only three of the ten most common letters are on the 'home row'.** Once again, these problems have been solved by alternate keyboards.

I.1.1.2 Not mobile

Keyboards are stationary and hence not able to be used whilst standing or on the move– and it is non-obvious how the keyboard could be altered for this problem to be overcome. Once again this is overhang from mechanical typewriters, for which portability was not a credible design goal.

I.1.1.3 The two interface problem

> **Two Interface Problem** The bifurcation of the current computing paradigm between desktop and mobile computers. As both have insurmountable domain superiority –in throughput and pervasiveness respectively–, users have to deal with two different interface paradigms rather than one *universal computing interface*.

As we have already seen, keyboards are presently irreplaceable because they are the highest throughput form of input. But mobile computing (§G.4) is also non-negotiable as a feature of many modern computing environments sue to its pervasiveness. This has created a fundamental bifurcation of our computing landscape between desktop and mobile computers. The present status quo is that desktops are used for 'serious'

[5] This innovation was made redundant with the invention of the electronic backspace.

[6] Again, a legacy of housing the mechanics of the typewriter.

computing whilst mobile touchscreen devices (§I.5) are better when on the move, with laptops providing some bridge in functionality.

To many, this may not seem significant, however we think this is extremely costly. Users have to deal with two quite different interfaces paradigms (different abstractions, interaction models, and even operating systems) to achieve essentially the same functionality. This surely has a cognitive overhead (§2.4.2) for users. Meanwhile, developers have essentially built everything twice at the OS/UI level. For example, in building some application you must choose to 1) build everything twice, 2) only build for one paradigm– leaving the functionality unavailable on the other, or 3) build some compromise– which is most often just (2) with a non-native interface provided to the second environment. Doing (3) sufficiently well is at least as hard as doing (1) in most cases so has little gain in efficiency. This leads to there being many basic computing tasks from one paradigm that simply cannot be well performed on the other. Consider the myriad of desktop programs and websites that do not have good mobile alternatives, or the great mobile apps for which there is no desktop counterpart.

Note that industry (noteworthy mention to Apple) has gone to great lengths to improve this situation, making interfaces feel streamlined and unified across desktop and mobile devices. Nonetheless, we think this equilibrium is expensive and are confident that we will see a simplification in the next interface revolution (§H.3) with the unification of high-throughput and high-pervasiveness in a single universal computing paradigm.

I.1.1.4 Moderate throughput

We hold the moderately controversial opinion that although the keyboard prevails for it's superior throughput (30-120wpm; average ≈40wpm) compared to mobile devices (20-40wpm), it is still far from the theoretical throughput limits of BCI (§1.3) (≥350wpm). **Consequentially, we think keyboard throughput is a bottleneck that can be dramatically improved with the reach existing technology. This is critically important because we rate throughput (§2.2) as the most consequential attribute in determining the power of a BCI interface device– any meaningful increase would be revolutionary. An important corollary is that, any new input device designed to replace the keyboard must have at least as high throughput to be viable.**

Note that we regularly encounter the common misconception that you can't (and wouldn't want to) type considerably faster than current keyboards allow (§J). For one, there are already devices such as the stenotype (§I.4) that comfortably allow input over 200wpm. We also think people tend to underestimate the true cost of low throughput input (§H.2.2.1.1).

I.1.1.5 The 3 hands problem

See The 3 hands problem (Sec. I.3.2).

Figure I.4: Selection of DIY ergonomic keyboards. Clockwise from top left: Santoku 2, Chordie, Grabbity Gloves, Dactyl Manuform, Charybdis, Chunky v2 ([14])

I.2 Ergonomic Keyboards

Many keyboards have now been build that deviate from the standard Qwerty layout for the purpose of improving design, particularly ergonomics. These vary greatly in their degree of change from the standard layout, from simple keymap changes (such as DVORAK and COLEMAK) to radical redesigns. We discovered that many users are drawn to ergonomic keyboards after suffering from RSI.

Ergonomic keyboards are made by companies but also (importantly) by enthusiasts (see Figure I.4). Fascinatingly, the cutting edge of this field currently resides in the open network of online communities dedicated to designing, building and sharing experimental interface designs. This network conducts a huge amount of specialist expertise, high quality documentation, open source designs and thoughtful discussion. In particular, we benefited greatly from Reddit's r/MechanicalKeyboards [37] and r/ErgoMechBoards [14] , and Discord's ZMK and ARTSEY communities.

Interestingly, handheld/wearable ergonomic keyboards have seldom been attempted, assumably for their added difficulty. One user strapped the two half's of a split ergonomic keyboard to their legs, another made a handheld model out of clay. Elsewhere on the internet we found a couple of old blogs discussing similar ideas [7], and numerous un-ergonomic TV-remote like controllers that made no serious attempt to match the keyboard for input bandwidth [45] . Most convincing was the mid 2000's AlphaGrip Controller (§I.6).

Though many ergonomic keyboards have been produced at commercial scale, none has yet challenged the popularity of the standard Qwerty layout. If you believe, as we do, that a suitable ergonomic keyboard is objectively better than the standard Qwerty keyboard, the question arises why they have not received greater adoption. We think this mostly due to switching costs and network effects (§F.3.1). Qwerty is very deeply

entrenched (every laptop comes with qwerty, everyone already knows it, etc.) so it will take a device that is significantly better and/or a radical change of circumstance to displace it. We think the next interface revolution (XR) (§H) is certainly such as circumstance, as the forcing function of pervasive output will render the stationary keyboard useless inviting a replacement bringing the smaller cumulative advancements of ergonomic keyboards with it.

I.2.1 Progress: problems solved by ergonomic keyboards

We will now overview the progress of ergonomic keyboards with respect to the aforementioned flaws of the Qwerty keyboard in the context of our 5AA framework (§2). We consider the class of problems solved by ergonomic keyboards as a whole even though mileage may vary greatly between keyboards.

Fully fixed problems from the Qwerty keyboard are the *addition of thumb clusters*– to make use of our most versatile finger, and the *elimination of RSI* which we are informed is suitably alleviated by measures including of 1) changing the keymap to have the most common keys most easily accessible, 2) changing the shape of the keyboard to better fit the natural hand (and provide wrist support), and 3) remove/minimise the most straining movements– making the layout ortholinear, reducing the number of physical keys, and moving modifiers away from the pinky (to the thumb).

Partially fixed from the Qwerty keyboard is *the 3 hands problem*– some keyboards are equipped with pointing devices and one-handed keymaps do exist, though are not especially popular, and transcending *moderate throughput*– there is the stenotype (§I.4) and other whole word (chorded) input methods which can achieve over 200wpm, and **we think there are more throughput-increasing techniques to be explored such as syllable input, strong autocomplete and mixed letters/whole-word input systems.**

I.2.2 Remaining problems

Remaining problems that ergonomic keyboards have not yet at all solved are *mobile pervasiveness* (§2.5.2) and *spatial computing* (§H.3).

I.3 Computer Mouse (Pointing Device)

The computer mouse was invented by Doug Engelbart at the Stanford Research Institute (SRI) in 1963-64 and famously demonstrated in 'The Mother of all Demos' in 1968. It was the first computer *pointing device*, designed specifically for the GUI computing revolution[7] (§G.3) to spatially control a pointer (*cursor*) with continuous and multi-dimensional data in a 2D plane– input that was not native to the existing keyboard (see Symbolic input vs. pointing devices; §3.1). Generally, in addition to moving a cursor, computer mice are equipped with one or more (symbolic) buttons to allows operations such as making a selection or inspecting the environment[8].

[7]The GUI also having been recently invented at Engelbart's research lab.

[8]Mice are also equipped with a *scroll wheel* by custom.

Figure I.5: Selection of computer mice. Clockwise from top left: Engelbarts orginal mouse, inside of classic mechanical mouse, gaming mouse with many buttons, ergonomic mouse, 3D mouse

The mouse was shown to Steve Jobs in 1979 who was so inspired that his company Apple scrapped their existing plans and redesigned everything around a mouse operated GUI. Subsequently, in 1984, the Apple Macintosh became the first consumer computer to popularise the mouse, and from there, the mouse (and descendent pointing devices such as the trackpad and touchscreen) quickly became a ubiquitous input paradigm, arguably eclipsing even the keyboard in popularity.

I.3.1 Alternate pointing devices

Similar to keyboards, *ergonomic mice* exist and are designed for optimum comfort and to reduce strain. Also notable are *gaming mice* which have been thought-provoking for their many additional buttons and *3D mice*, which generalise the pointing device to at least three degrees of freedom, making them natural fit for the next interface revolution: XR computing (§H), which is innately three dimensional. Alternate pointing devices have also been invented and are highly popular, including; the trackpad– which is standard on laptops, the joystick– common on game controllers, the touchscreen[9]– which dominates mobile, and eye tracking (§3.3)– which we bet is the future. In out practical project we explore the benefits and drawbacks of each (§K.6).

I.3.2 The 3 hands problem

Engelbart quickly noticed a usability problem with his newly invented mouse as an accompaniment to the existing keyboard. This was that a keyboard requires two hands to operate, a mouse requires one hand to operate, but humans only have two hands– leaving us one hand short of the three hands needed to operate a keyboard and mouse simultaneously. We refer to this as *the 3 hands problem*.

[9]Specifically a subset of touchscreen functionality.

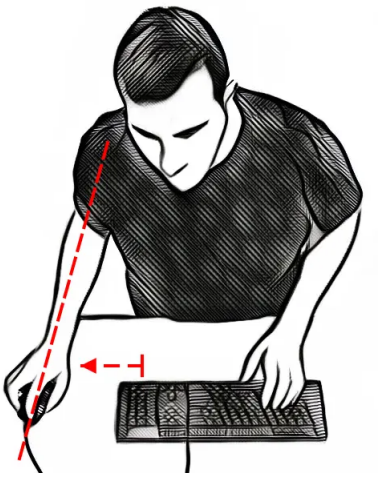

Figure I.6: The 3 hands problem ([54])

The 3 hands problem The impediment that it takes three hands to control a standard keyboard and mouse simultaneously.

The 3 hands problem means that using standard keyboard and mouse together incurs a switching cost when swapping between devices, i.e. the time, effort, and strain of picking up your hands, moving and putting them down, and finally re-calibrating their position– all before making the intended input (and then back again if the switch was transient).

Evaluating through our 5AA framework, we identify significant *physical obstruction*, increased *latency*, and reduced *throughput*, all in our opinion, unnecessarily. This limits our ability to use the two devices together, this is particularly problematic for activities that are both keyboard and mouse intensive such as CAD, gaming and text editing. A partial coping mechanism has evolved in which the most important keyboard controls are usually stacked on the left hand of the keyboard freeing them to be used simultaneously with the mouse. **In extreme cases, users forgo the native input of either a keyboard (symbolic) or mouse (pointing) all together, unnaturally pigeonholing one's functionality into the other because the switching cost is so high.** For instance, programmers are infamous for avoiding the mouse. They do so not because (as some claim) the mouse is a bad input paradigm, but because you can't use one simultaneously with a keyboard, so it becomes optimal to move as much functionality as possible to the keyboard, sacrificing the genuine value of the mouse. For example, the optimal way to navigate and select text is with a pointing device, yet many programmers find symbolic keyboard methods such as 'vim bindings' faster. CAD mice exhibit a similar behaviour in reverse, an unholy amount of symbolic buttons.

The obvious conclusion is that any new input device should better solve the 3 hands problem, by making the entirety of keyboard functionality available from just one hand and/or by integrating a pointing device ergonomically into the rest of the device. Engelbart realised this and invented *the Keyset*, a one handed keyboard

replacement, but it never caught on.

I.4 Stenotype

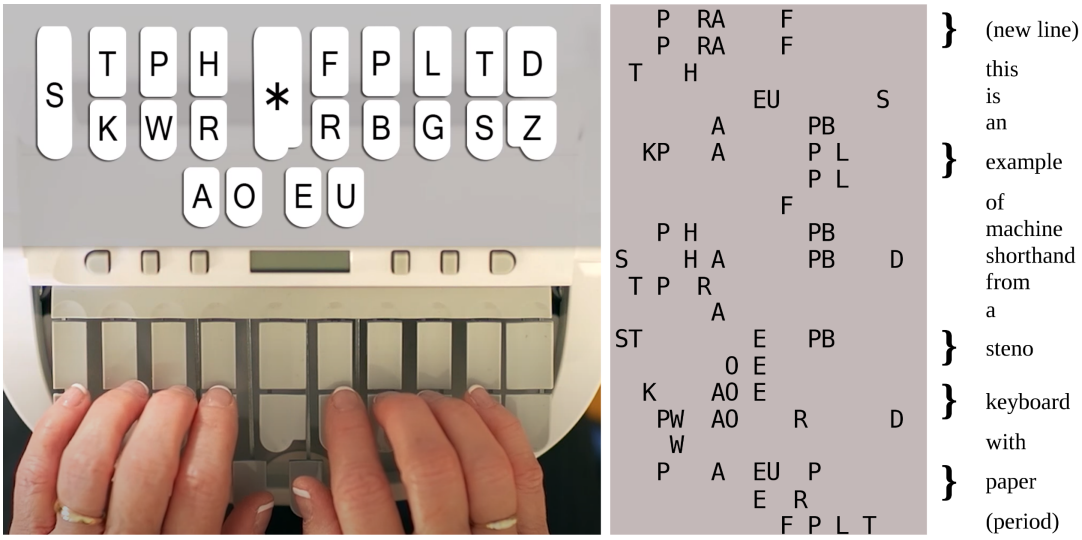

Figure I.7: Stenotype machine and keymap (left) and example output shorthand (right)

A *stenotype* (machine) is a specialised chorded keyboard used by steographers[10] to keep up with speech, 180-300wpm. The stenotype machine works by typing in syllables[11] rather than letters, and each word is a unique 'chord' of one or more syllables pressed and released simultaneously. For example, the word calendar, which takes eight strokes on a normal keyboard, takes just one three syllable chord ("cal" + "en" + "dar"). Note that the onboard computer adds spaces and interprets words. In this condensed for stenotype machines have only 22 keys (unlike the standard keyboards which has 70-105) which combine to make hundreds of syllables. Additionally, the steno machine uses ortholinear (non-staggered) keys with a light activation force, making it easy to press multiple keys at once.

One drawback is that words have to be in a preexisting dictionary which must be memorised. This means it takes a very long time to become proficient, and makes it unsuited to some use cases such as programming. Overall, stenotype machines show that it is possible to type much faster and with fewer keys than a standard keyboard using chords of syllable input.

Note that until their recent digitalisation, the output shorthand needed to be manually translated back to full English, cancelling out any overall time benefits. Now with computers this is no longer needed.

Figure I.8: Selection of touchscreen interfaces. From left: smartphone, tablet, smartwatch. Some touchscreens are a better fits for their form factor than others.

I.5 Touchscreen

A touchscreen (input device) is an intuitive *pointing device* often equipped with a reasonable (symbolic) 'on-screen' keyboard and 'gestures'. Touchscreens are the defacto choice for portable *mobile* devices chiefly due to their pervasiveness– they come in many sizes so can comfortably be taken anywhere; unlike a keyboard & mouse. More generally, they are favoured for their simplicity, ease of use and good-enough symbolic input (that is not obstructive when out of use).

I.5.1 Advantages

One factor contributing to their ease of use is that they enable the user to interact directly with what is displayed– mimicking the real world, where your inputs are fluid with the environment (outputs). Relatedly, the interface is very *discoverable*– and almost all functionality is visually hinted at with few hidden commands, unlike a keyboard which has many shortcuts that do hidden things across many applications. Additionally, touchscreens (also trackpads) are notably better than a traditional mouse at gesture based symbolic input (e.g. pinch to zoom) which can often be highly intuitive. Note that mobile touchscreens semi-solve the 3 hands problem (§I.3.2) by essentially moving the keyboard into the pointing device. Compared with the keyboard & mouse, touchscreen GUI interfaces tend to require larger buttons as fingers are less precise than a mouse[12], this is especially notable on smartphones where the screen is small and buttons are very large relative to the screen– essentially making smartphone/desktop GUIs incompatible.

I.5.2 Drawbacks

Mobile touchscreen input does have its drawbacks, it is generally slower (*throughput*), less accurate[13] not *tactile* (so cannot be operated without looking) and more *obstructive*,

[10] A person who's job it is to transcribe speech in real time (generally shorthand).
[11] A specialised set of syllables.
[12] And impede visibility.
[13] Than pointing devices such as the mouse or stylus.

requiring constant gaze, screen real estate, and focus. **This means the touchscreen is not the preferred choice for the most demanding work.** As a thought experiment on the limits of touchscreens, imagine telling a fighter jet pilot that you will soon send them to battle with all their controls replaced by touchscreens.

I.5.3 Prevalence (vs. keyboard & mouse)

Touchscreens are increasingly becoming a favoured input paradigm over the keyboard and mouse by users and interface designers for their perceived simplicity and generality[14]. Many casual computer users have only a touchscreen device. Moreover, consider the 'iPadification' of the MacOS desktop operating system (bigger buttons, larger text, etc.), despite the fact it does not (yet) have a touch screen.

I.5.4 The mistake of miniature physical keyboards

Miniaturised physical keyboards have been tried on mobile devices but they are now relatively uncommon as they provide a negligible benefit. Compared to a an on-screen keyboard, they take up space– both physical and on-screen real estate (which can be precious for mobile devices), are not truly tactile– as they are too miniature to be operated without looking, are not meaningfully familiar to keyboard users– requiring new muscle memory, and are less flexible– consider emojis and foreign language input. The notable exception is larger touchscreen devices, such as tablets, where attachable keyboards are popular, note that these are full-size keyboards that do no require learning an essentially different input device.

I.5.5 XR & High-Pervasiveness

When projected into the coming XR revolution (§H), or for that matter very large display panels, the 'direct on environment' input of touchscreens could become cumbersome and inefficient in such a large environment, ideally you want some projection from a smaller input space to the full environment (like how a touchpad is smaller than the display it navigates). Additionally, in high-pervasiveness environments, such as those involving movement, a touchscreen can be unergonomic, such as on the Apple watch (see Figure I.8). This is one reason touchscreen smart watches have not been as dominant as touchscreen smartphones.

I.6 AlphaGrip Controller

AlphaGrip is a handheld ergonomic game-console shaped keyboard controller. It was created in the mid 2000's, enjoyed limited commercial success but fizzled out as the ownership neglected evolving the project. Of all the input devices researched, AlphaGrip is the device that is most in the spirit of our practical work, completing the most of our design goals (§5), being *pervasive*, *functional*, *ergonomic*, *tactile-ish* and *fast-ish*.

[14] As previously stated, we think this falls short of full generality.

Figure I.9: AlphagGrip ergonomic keyboard controller

We were able to get our hands on one for testing, noting that it lacked good keyswitch ergonomics, wireless capabilities, good modifier positions, but otherwise was functional and highly usable. After some practice we were able to type at 30wpm on it, similar to our one handed HCID v0.3. We feel that even a modernised rendition of the AlphaGrip would be commercially viable. Nervous System Interface (NSI) ?

I.7 Flight Joystick

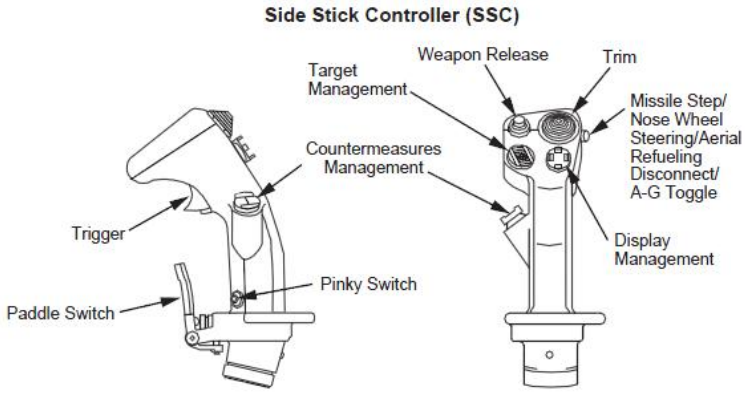

Figure I.10: F-16 flight simulator joystick

The computer simulator joystick (and throttle) is directly modelled on the fighter jet joystick used to control the navigation and essential functionality of the aircraft. Notable ergonomic features include it's acclaimed superior 'trigger grip' and extensive thumb cluster of functionality.

We found the shape and layout of buttons convincingly more ergonomic that all other devices researched. We suspect that the high velocity, notoriously difficult 3D controls, and competitive military environment has evolved highly optimal solutions

that apply well designing input controllers for a 3D XR computing environment. Note that our study of the flight joystick came after the design of our latest prototype so its influence is not yet represented in our practical work.

Interestingly, the aeroplane joystick did not used to have buttons, they were a 1950's British military innovation, acronymed *HOTAS* (*hands on throttle-and-stick*), to allow pilots to perform all vital functions whilst keeping their hands on the navigation controls of the aircraft. Previously they had to switch between navigation and operational functionality which could be costly. We think this is almost perfectly analogous to our 3 hands problem (§I.3.2), suggesting that pointing devices (mice) and symbolic input (keyboard) should be similarly unified in a single interface to improve performance.

Appendix J

Common Misconceptions (why nobody is building this)

So far we have come to a strong conceptual case for a new Hand Controlled Input Device (HCID) to supercede the typewriter keyboard (and mouse/touchscreen) based on its existing flaws (§I.1.1) and unviability in XR; an interface revolution we think is near, and for which there is incredible opportunity.

Accepting this raises the question of why nobody is building such an input device.[1] We reason that, for there to be such a distinct lack of interest, in what seems to us straightforward and important, either our logic must be flawed, or there are some systematic reasons this is being overlooked.

We cover some common misconceptions that we have eluded to throughout different sections.

1. *Base-rater's fallacy (the future will resemble the present).* This form of *status quo bias* is the psychological proclivity for people/society to overlook *exponential tech–* technological improvements to the status quo with a rapid (exponential) rate of change, assuming (incorrectly) that things will roughly stay as they are. We assume things don't change $\rightarrow$ then they suddenly do $\rightarrow$ then we quickly forget that things weren't always as such. This psychological bias animates many of the following more concrete fallacies. For a specific example, see Moore's Blunder (§E.4.1).

2. *XR isn't coming (soon).* Many argue that XR computing is either not: viable, useful or imminent. In direct opposition to the (never coming) misconception, we have provided a strong theoretical framework (§2) which models how XR offers an irresistible upgrade to the power of humans through their computers. With respect to the aforementioned base-rater's fallacy, notice how we marvel at how much mobile computing have changed our lives, yet generally do not seem to expect another similar revolution. On timeframe concerns, we have developed a theory for why this revolution hasn't been possible in the past (§G) and tracked

[1] Note the exception of a couple of similar solo projects [33, 46] who we are in contact with.

its progress to the present point of imminent breakthrough (§G.5). Also note that we think our envisioned input device has many use cases that are independent of XR (§6).

3. *The iPadification Fallacy*. This relates to a collection of related endemic (yet generally assumed and unspoken) fallacies thematic on the erroneous expectation that future computers are (only) increasingly like iPads –or rather mobile computing more generally–: simple, locked down, touchscreen first/only, etc. Fundamentally this overlooks the value of symbolic input (§3.1) and the related good of high throughput (§2.2). This fallacy is encountered in various different forms.

   (a) *Users want maximally simple interfaces*. This neglects the fact that simplicity trades off with functionality– which users also want.

   (b) *Text/command based (symbolic) input is unessential*. Looking at market leaders in XR (particularly Meta), we infer they consider text input unessential as they make no serious attempt to develop it in their gaming/entertainment XR metaverse (see Figure 6.1). This idiocy is even explicitly expressed, here by market leader Qualcomm: "While [A.] immersive XR applications are often less oriented towards text capture, there are still [B.] (few) use cases where some sort of keyboard input is required." [36]. May we suggest that XR applications are less oriented towards text capture in no small part *because* they haven't been provided with a workable text interface[2].

   (c) *Mobile > Desktop*. This view ignores the drawbacks of touchscreen input (§I.5.2) and the reciprocal merits of keyboard & mouse operated desktop interface (§I.1). Just because the mobile revolution followed the desktop, it doesn't mean that future development will be exclusively in this direction (rather than merging the best parts of both). If mobile-esqe devices are the future, why are you still using the desktop?

   (d) *They want to keep you stupid*. This is the slightly conspiratorial view that large technology companies are (consciously or unconsciously) building interfaces that exsessivly limit users degrees of freedom. This could be for a number of reasons: it makes their jobs easier, it locks users into their network, it makes them more money and users do genuinely value simplicity. The net effect it gives them more power and makes us less capable.

4. *Mobile entrenched the desktop*. We have a subtle theory that the mobile revolution entrenched the status quo bias of the desktop. In not fully displacing the desktop it created the illusion that the desktop could not be displaced in a subsequent interface revolution. This lead to things such as a decrease in research funding for new computing interfaces.

5. *Voice Assistants will take over*. See Voice Control (§3.4).

6. *NSI is coming soon*. This is the relative overlooking and underfunding of XR (the near term horizon) in favour of NSI (the far horizon). It's not that we think NSI

[2]Their argument is standard case of *confirming the consequent*; A, then B; B, therefore A.

never, but accept that its a much more complex and hence longer term technical challenge. We think this stems from a lack of imagination and appreciation of how revolutionary XR will be to general-purpose computing (§G.5) in the near term. This is the equivalent of not learning to drive in 1900, because you're holding out for driverless cars.

7. *Keyboards are good enough (you can't, and wouldn't want to type faster than they will allow.)*. Many severely underappreciate the flaws of the keyboard (§I.1.1), don't consider the true cost of low throughput input (§H.2.2.1.1), and can't imagine the opportunity for something radically better (§5), even ignoring the advent of XR.

Appendix K

Design Decisions and Discussion

The following is a (lengthy) retroactive record and discussion of the design decisions made up until this point (HCID v0.3). We have tried to conduct discussion in the order of logical dependency rather than chronologically, such that in retrospect we can see that every decision is approximately dependent on those which came before it. Of course, in reality, the decision-making process was much more chaotic.

Key

☒ **Chosen** ☐ ~~**Ruled out**~~ ⊟ **Maybe/In progress**

+ positive aspect	- negative aspect

K.1 Portable keyboard vs. EMG wristband

1. ☒ **Portable Keyboard** envision a handheld with lots of buttons.

+ existing knowledge applicable	- hands not free
+ tactile feedback	- not fully wearable
+ fast MVP and iteration cycle	- limits range of hand movements

1. ☐ ~~**EMG Wristband**~~ envision a smartwatch that can read hand gestures.

+ full range of hand movements	- much complex engineering
+ hands are free	- far away MVP
+ fully wearable	- tactile feedback very hard
+ more aesthetic	- ADC hard

(1) was chosen, despite (2) being considered a more ideal final form. In fact, we did spend some time trying to work on (2) but it was considered too ambitious, see HCID Wrsitband Concept (Appendix B.1).

(1) was considered far more practical to build as much of the *vast existing knowledge can be applied* that comes from the custom keyboard building community. Also, the

benefit of *tactile feedback* is very important (and easily underestimated)- it provides positive/negative reinforcement for learning and can transform the fengshui of a device. Additionally, when compared with (2), we believe it is much simpler to build working prototypes, specifically ones that cross the threshold of being good enough to actually replace a keyboard/mouse resulting in a *fast MVP (Minimum Viable Product) and iteration cycle* which are very powerful facilitators in product development. One serious drawback of (1), especially compared to (2) is that your *hands are not free* to do other things when in use or between uses. A related disadvantage of (1) is that it is unlikely to be fully wearable (pervasive), you will have to take it off regularly to perform other tasks.

(2) allows for a more *full range of hand movements*, such as natural gesture-based ones, rather than just those which correspond to button presses. Additionally, your *hands are free* to hold things when not in use, making the device *fully wearable*. The minimalism of a simple wristband makes the design more *aesthetically pleasing* as it is almost unnoticeable and there is nothing to carry around. Unfortunately, this project was considered too much work for an undergraduate part-time project (perhaps even too much work for a moderately sized team). Some of the *complex engineering challenges* include: novel wearable hardware, electrode fabrication and/or optimisation, signal processing/filtering, an ML generated Hand-Pose-Network and gesture-command mapping. These make a working *prototype/MVP very far away*, making it hard to test and improve the device.

Despite these difficulties, we maintain that (2) is a *very good idea* and would like to work on it in future. We predict that within the next 10 years EMG-Wristbands could become humanity's predominant input device. I think the critical problem to resolve is *tactile feedback*– most likely implemented as targeted electric stimulation.

Finally, we want to mention that a lot of the design may be transferable from (1) to (2), many of the constraints relating to the human will be the same and in particular, the keymap could be transferable directly, as from guitar to air guitar– the same commands, just in open space. It's also worth noting that (1) and (2) could technically be used simultaneously.

K.2 One-handed vs. two-handed

1. ☒ **One-Handed** envision 2 VR controllers.

+ flexibility	- potentially slower

2. ☐ ~~**Two Handed**~~ envision a standard game controller with lots of buttons.

+ potentially faster	- no free hand

It was decided that the device should be operable using only one hand. This solves The 3 hands problem (§ I.3.2) providing *flexibility*, allowing the device to be used in conjunction with doing other things such as using a mouse, holding a cup of coffee,

drawing with a pencil, etc. The potential *speed lost* could be made up by having the optional ability to use both hands.

This raises the additional question of how to implement the optional second hand. It's possible to either, (a) symmetrically mirror the functionality of the first hand, or (b) provide asymmetric functionality. We had a strong intuition to choose (a), as we were drawn to thinking of ambidextrous symmetry as a benefit. It is also worth noting that (a) is easier to implement as you only need to design and program a single keymap. Perhaps this could be paired with the option to implement some kind of semi-permanent layer on the second device giving the de facto behaviour of (b).

There is then also the question in the two-handed case of whether the two halves should act (c) separately or (d) as a whole, e.g when pressing Control, should it act globally or be bound to the side it was pressed on. We chose (c) because it was easier to implement but it might be possible to add the ability to switch between the two.

K.3 Handheld vs. leg-mounted vs. wrist mounted

1. ☒ **Handheld** something you hold in your hand.

+ familiar/obvious + 3D-Ergonomic layout possible	- probably requires strap - peoples hands differently sized

2. ☐ ~~**Leg Mounted**~~ strapped to your leg (see [40]).

+ 2D-array layout possible + trivial to implement	- cannot move arms - unnatural placement on legs

3. ⊟ **Wrist Mounted** think Assassins Creed blade.

+ stable support + 2D-array maybe possible + super cool	- tricky mechanical engineering

(1) was chosen primarily because it offers more freedom of movement than (2) in which you *cannot move you arms* which are *unnaturally placed on your legs*. (3) was only seriously considered after building (1) and is a potential avenue to be explored in future.

One drawback of (1) considering our decision for a one-handed design, is that we will now almost certainly require a handstrap which will make the device more cumbersome and could be considered ugly. Finally, there is the consideration that *people's hands are differently sized*. It was reasoned this could be solved by first designing a device solely for some individual hand-shape and iterating from there; either to a universal one-size-fits-all or to multiple sizes or even individualised design[1].

[1] Perhaps use hand scanning to generate a parameterised casing to be 3D printed.

(3) is an interesting idea that was only come across recently and has not been ruled out for future designs. Taking inspiration from the hidden blade in the video game Assasins's Creed it has the advantage of obviously being super cool. More technically, it has the advantage of hopefully acting as a *stabilising support* to the device, which turned out to be a serious problem for (1), and makes a *2D-Array (QWERTY) layout* possible, although you working within a somewhat limited space.

K.4 2D-Array vs 3D-Ergonomic

1. ☒ **2D-Grid** like a standard keyboard. Anything that lays keys out in a grid.

+ spatially organised + one size fits all	- potentially unerogonomic

2. ☒ **3D-Ergonomic** buttons can be anywhere on the surface of a 3D object.

- ergonomic	- harder electronics - people have different shaped hands

A hybrid was chosen; (1) for the back (fingers) and (2) for the front (thumb).

My initial assumption was that (1) was universally bad and (2) would be much better as hands are not naturally grid-shaped. However, we came to appreciate (1) as it *spatially organises* the functionality and is *one size fits all*– working for all hands. Additionally our research into hand movements produced the FBD Model (§ 4.1) which imposes a grid-like symmetry between fingers.

K.5 Key Switches

1. ☒ **Mechanical Key Switches** loved by keyboard enthusiasts.

+ ergonomic + standardised	- fixed large size - no FBD - expensive

2. ☒ **Membrane/Rubber Dome Switches**. standard switches on a game controller or TV remote.

+ variable size	- relatively poor ergonomics

3. ☐ ~~**5-way Nav Switches**~~ specialist 4+1 directional switch [42].

+ FBD compatible	- very bad ergonomics - high activation force

4. ⊟ **3-way FBD switches** our own design in which you can push a rocker Forward, Backward or Down.

+ FBD compatible	- could not find

5. ⊟ **Other Switches** rotary encoder, D-pad, rocker, trigger, gearbox, etc.

+ whole array of options + spatial computing	- niche

Our starting point was our FBD model (§ 4.1) so we set out to find an accommodating 3-way switch. We envisioned a lever/paddle that could be pushed forward, backward and also down. Our research uncovered many weird and wonderful switch types, but unfortunately no 3-way rocker.

We did find (2), and built our second major prototype HCID v0.2 (§ B.3) using one for each finger. However, the ergonomics of (2) turned out to be terrible. Chiefly, the switches had a very *high activation force*, particularly on the 'down' press. This was bad when pressing a single key, but as chords were essential to our design, this made it prohibitively straining to press multiple at once. Additionally, the travel distance was very short[2] and pressing the switches was very 'clicky' and did not feel good. This can sound superficial but it made the device unpleasing to use.

We were reluctant to give up (2), as it was the only way we had of implementing our FBD model (we also figured FBD gave me three 'rows' which would be enough to implement a QWERTY like layout layer, fulfilling our *isomorphic* design criteria). We considered using four game-console joysticks instead but found this to be quite daunting/impractical as they are large analogue devices.

We considered attempting to build specialist hardware to achieve (4) and fulfil our FBD model (perhaps attaching a 2-way rocker switch to a standard key switch in some way). However, with limited time, hardware expertise and resources, we decided to leave it and move on, without completely ruling it out for future experiments.

> **Update** We have recently found some new leads on (4), by searching for 'multi-directional switches'. This will be investigated for future prototypes.

As such, we switched to our current choice of (1). Having never used a mechanical keyboard before, I discovered their key switch *ergonomics* really do feel great. Additionally, being *standardised* (and familiar to the keyboard community) was a huge advantage in being able to utilise existing resources.

[2]We somewhat fixed this on the Forward/Backward motion attaching a 3D printed lever to extend the length of the navigation stick.

(1) however does come with some disadvantages. Firstly, they come in a *fixed large size*, which fixes the number you can fit in a space. Critically, this reduced us from our favoured 3-keys per finger to 2-keys per finger in our latest prototype[3].

We decided against using (2) in the general case, despite their *variable size*, they generally have much less satisfying *ergonomics*, feeling a bit tacky. However we did using (2) for some smaller, less prominent buttons such as the pause key.

(5) was considered. Perhaps at some point, we would like to add a rotary encoder for scrolling and some game console triggers for shooting the bad guys.

K.6 Pointing Device (Mouse)

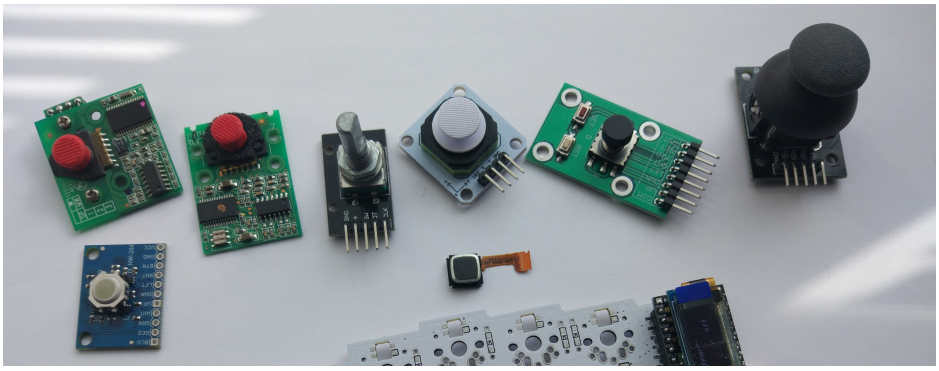

Figure K.1: Selection of pointing devices ([18])

1. ⊟ **Joystick** as seen on most game controllers.

+ game controller compatible + can be utilised for symbolic input + natural for navigating 3D space (VR)	- poor control

2. ☒ **Small Rollball** Blackberry style.

+ good control + compact + easiest to implement on ZMK	- not very best control - no continuous scroll

3. ☐ **Large Rollball** as seen on some specialist mice.

+ best control	- large - no continuous scroll

[3] it might have been possible to squeeze three rows in, something to experiment with in future.

4. ⊟ **Trackpoint** 'nipple' classically found on Lenovo Thinkpad keyboards.

+ good control	- no inbuilt button

5. ☐ **Trackpad** as found on every laptop.

+ familiar	- no continuous scroll

6. ☐ **Pointer** as seen in VR controllers.

+ natural	- unstable for fine movements - strenuous to hold up

7. ☐ **Eye tracking** point-and-click with look-and-blink/think.

+ the future	- not here yet

We decided that this decision is not particularly urgent to finalise as it is reasonably easy swap pointing device between prototypes and their functionality generally acts in isolation from the rest of the device. Note that all options are viable and many people have a personal preference. It potentially might even be possible in production to let users choose their own or have multiple such as in the Steam Deck [41] which uses both (1) and (5).

(1) has the advantage of being *game console compatible-* (most likely) necessary for implementing the functionality of a game controller, which could remove the need to switch between a controller and keyboard device. Additionally, we think it could be *utilised in symbolic input-* using large movements to control things like modifier keys or layers. (1) also feels the most *natural for navigating 3D space*[4]. The downside of (1) is that is probably the only option that has *poor control* for fine movements– this is probably a deal breaker for sole use as a pointing device. (1) in combination with a second pointing device (we don't currently have any strong preference which), is probably our idealised option. Note, earlier prototypes used (1) (see HCID v0.2 (Appendix B.3).

(2) was used for the most recent prototype as it was *easiest to implement on ZMK* (our chosen firmware) due to having the best support.

(3) does come in a range of sizes and many swear by (4) having the best control. (5) despite being *familiar* (ubiquitous on laptops), does have the disadvantage of *no /continuous scroll-* the ability to, say on a joystick, hold/lock in a maximised position, as opposed to having to pick up and replace your finger when you reach the end of the trackpad. (6) has the advantage of feeling very *natural* see Spatial computing (§ H.3) however has the disadvantages of being *unstable for fine movements* and *strenuous to hold up* for long periods of time. (7) will certainly be *the future of pointing devices*, perhaps sooner rather than later,but suffers from not currently existing in an easy to integrate manner.

[4] this could potentially be extended to 6-DOF

K.7 Button Layout

What layout of physical buttons do we want available for each finger (and thumb) to reach? To recap, we have previously established a preference for 1) a 2D-Array for the fingers and a 3D-Ergonomic layout for the thumb cluster, 2) mechanical key switches as the main button type, 3) a small thumb operated trackball as a pointing device (for now), and 4) ideally 3-buttons/per finger (FBD). We also think the following should be added: 1) a power button to turn the device on/off, and 2) a pause button to lock keypress' from being registered.

For our most recent prototype (HCID v0.3) space constraints made us settle for only 2 buttons per finger, giving us a 2x4 2D-array for the fingers. For the thumb cluster, we designed the casing with 6 additional thumb buttons (3 on either side of the trackball), but only ever implemented two (one on either side), before wanting to move on to the next prototype.

K.8 Casing

In early prototypes (up to HCID v0.2; § B.3) serious consideration of ergonomics was skipped in favour of a cuboid (resulting in a quicker first prototype).

We the decided to first design the casing to fit a single hand before trying to generalise beyond that. Even so, completing our Design Goal (§ 5) to make the device ergonomic was much more difficult that expected.

Firstly, CAD is not our specialist subject, so there was a lot to learn to build something that wasn't just a glorified box. Additionally, unlike software, which generally has immediate feedback, the iteration cycle (in theory) for 3D printing our casing was roughly 7 hours to 2.5 days (depending on the machine/settings). In reality, our models would rarely print correctly (see fig. K.2) which made the true time-to-print many times longer. Of the entire project, we spent maybe 40% of the total time on the casing, and the majority of that was spent fighting the 3D printer. Combined with Covid restrictions heavily limiting our access to the university's 3D printers, this became a serious bottleneck in the project.

Figure K.2: Casing misprints, Clockwise from left; warping, under-extrusion, contamination, warping.

We spent a huge amount of time redesigning the casing to fit comfortably in the hand, line the buttons up in the right place for the fingers, get the 'angle' between the thumb and fingers correct, and stop the device 'rocking' when buttons were pressed. We went

through dozens of prototypes including clay models and radically different designs (see Figure K.3) and still never arrived at a design we were truly happy with. In this sense the casing is very much a work in progress, however, we do think our design is going in roughly the right direction. For technical diagrams of the current prototype see Casing Technical drawings (§ 8.1.2). Important features of our design so far include 1) a large chassis to 'fill' the hand, 2) two holes to accommodate a hand strap, 3) a gradual taper such that the device is thinner to hold towards the pinky finger, and 4) a support 'wing' at the base of the index finger to help balance the device.

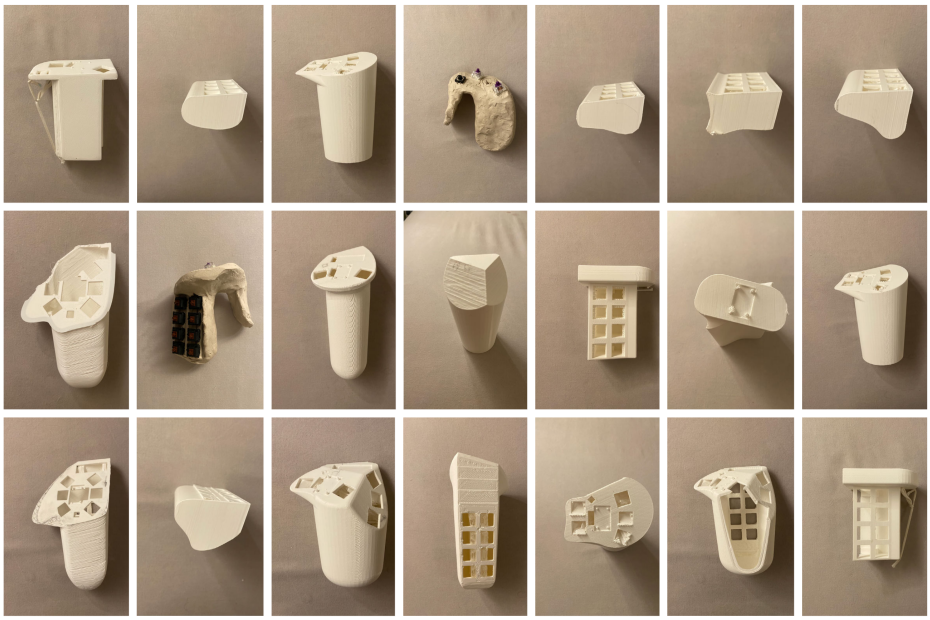

Figure K.3: Various casing prototypes

K.9 Firmware

1. ☐ ~~**Custom Firmware**~~ writing our own firmware, from scratch.

+ full control	- reinventing the wheel - time sink

2. ⊟ **QMK** [35] popular old-school custom keyboard firmware.

+ most popular + best pointing device options	- wireless is messy - licensing issues

3. ☒ **ZMK** [55] bluetooth-first keyboard firmware.

+ highly popular + wireless first	- pointing devices are bleeding edge

4. ☐ ~~**BlueMicro**~~ [9] Arduino based keyboard firmware.

+ arduino based	- least support - verbose chording mechanics

For far too long we attempted (1). The main reasons were: a) we didn't really understand keyboard firmware, and b) being a School of Informatics project, we subconsciously felt that we should take on the most ambitious programming project possible. This turned out to be a massive waste of time that cost us hundreds of hours we could have spent elsewhere to essentially build a buggy subset of existing open source solutions. Some of these projects have dozens of active contributors and all of them are tens-of-thousands of lines of code and built over multiple years. The silver lining of attempting (1) was that it taught how many things are implemented under the hood.

Customising and extending existing keyboard firmware was exactly what we needed. This allows the imported firmware framework to handle the low level hardware and provide abstractions whilst we worked on novel functionality. We eventually chose (3) over (1) and (2), primarily for its *wireless first* approach and supportive community. We originally ruled out (3) for not having pointing device support however later found that it does exist in an unofficial experimental git branch. (2) was discounted despite having the *best pointing device options* because its *wireless support is messy* and has open source *licensing issues*. (4) was briefly tried. Being *Arduino based* was an advantage for prototyping new features however it has the *least support* and the most *verbose chording mechanics* (which we intended to rely on heavily).

K.10 Microcontroller (MCU)

1. ☒ **nice!nano v2** [29]

+ ZMK recommended + relatively cheaper	- no motion sensors

2. ☐ ~~**Adafruit Feather Sense**~~ [4]

+ motion sensors included + arduino compatible	- not ZMK recomended - relatively more expensive

(2) was my original choice, but current prototypes were made with (1). Our main requirement was onboard bluetooth, which only gave us a handful of options, most of which are based on the same nRF52840 SoC [38].

(2) was originally chosen for its plug and play battery, inbuilt bluetooth, *Arduino compatibility*, and collection of *motion sensors included*- with the hope of using them for gesture support. However, choosing an MCU ended up being quite dependent on the

firmware framework. (1) was the most supported on our ZMK firmware, and gestures support was still a long way off. Technically (2) is supported by ZMK, and could be used in future but would be more work to get set up.

K.11 Keymap

This was by far the most intellectually difficult part of the project. Deciding how to input movements should map to commands is a very large search space– there's almost an infinite number of ways you can layout and combine keys and theres no obvious way to begin working out what is optimal. We realised early on that often the best you can do is guess and iterate.

K.11.1 Functionality goals

To recap, the important functionality we want to include is 1) the full range of standard keyboard symbols (letters, numbers, special characters, etc.), 2) mouse movement and buttons, 3) multi-letter/whole-word direct input, and 4) An isomorphism to QWERTY (and similar layouts)- this essentially means a 3x5 grid for each hand. For more see Design Goals (§ 5).

K.11.2 Constraints

The important decisions made so far constraining our keymap include 1) 2x4 Array of key switches[5] (potentially 3x5 in future)[6], and 2) 2 additional thumb buttons (with the potential to add joystick gestures in future) These constraints make the design space a lot more manageable[7].

K.11.3 Design

Note that Keymap Nomenclature (Appendix A), covers the following section's specialist terms.

The accepted wisdom of custom keyboard layout design goes something as follows. You have a *base layer* containing (in the most extreme case of having the fewest keys) *alphas*, *essentials*, some important subset of *symbols* along with *modifiers* and layer switching keys[8]. The logic here is you want the most frequently used keys to be most easily accessible. Say there are 7 essential symbols (`.` `,` `;` `)` `(` `:` `/`), and two layer switching keys (`LayerUp`, `LayerDown`). This gives us a total of $26+4+7+4+2=43$ keys to implement. Some arithmetic reveals that our current design has only $4\times2+$

[5]This prevents us from implementing a QWERTY isomorphism for this prototype.

[6]Note that as the button layout is subject to change in future prototypes, so is the keymap.

[7]This is far more true in hindsight than it was in the development process as the complexities of the keymap had to be strongly taken into consideration when designing the button layout.

[8]A slight variation of this is the Vim [50] keymap, who's base layer is commands based and the alphanumeric 'Insert' mode has to be entered like any other layer. This reveals the hidden (incorrect) assumption that computer input is foremostly alphanumeric, rather than command based. We intend experiment with this in future.

$2 = 10$ keys. From here the natural choice is to involve *chording*[9] Chords are used extensively in musical instruments (such as the piano) and there already exist successful *chorded-keyboards* such as stenography machinces. A quick sum of 1, 2 and 3 key chords on 8 keys (leaving the two thumb keys) gives us 92 key combinations, which is already more than enough.

From here we have the task of laying out our base layer. As we have already established, an isomorphism to QWERTY is not possible given out button layout. Generally, the perceived wisdom[10] for new keymaps is 1) to have the most common keys in the easiest to reach position, and 2) avoid the most straining movements (see gloss. RSI).

The most common letters in the English language[11], roughly in order of frequency are, ETAOINSRHLDCU [30]. From here there are many ways to start laying out keys and we have come to believe that the exact placement is somewhat arbitrary.

Amazingly, at this time we found the ARTSEY keyboard community [6], who had already designed a layout for a 2x4 one-handed keyboard with many of features we have been looking at. One additional feature they brought was using an isomorphism to letter shapes (similar to sign language for those familiar) as a secondary scheme for laying out letters (after frequency). For example, the chord for the letter `l` has the shape ☐ ☐ ☐ ☐ / ■ ■ ■ ☐ (left hand), which can feel natural and be easier to remember.

We decided to use this as a starting point for our keymap. Once we successfully ported ARTSEY to our hardware, there were some obvious minor matters. Firstly, it was designed to be used on a flat surface rather than in a handheld so some chords were strenuous (e.g. ones involving the middle and pinky finger). Additionally, we also had two additional buttons to add, which upon suggestion we mapped to two modifier keys (`shift` and `GUI`).

K.12 Wiring

1. ☒ **Hand Wired** Individually soldering components.

+ flexible	- messy and can become overwhelming - tedious: does not scale

1. ☒ **Printed Circuit Board (PCB)**. Laminated switch structure, compactly connecting wires.

+ ideal + faster to assemble	- additional step - not flexible

[9] alternatives such as *arpeggios* (*leader-keys*) are possible but seemed less natural.
[10] we did experiment with other things.
[11] It does vary slightly depending on the corpus used to do the analysis.

Our current prototype uses (1), but we are switching to (2) for our next prototype and have already started designing two PCBs, one for the back (fingers) and one for the front (thumb cluster).

(1) is great for prototyping, as you are *flexible* to do what you want. However, all the wires can quickly get *messy and can become overwhelming* and *tedious so does not scale* as it can require many hours to wire a single keyboard.

(2) is more *ideal*; all commercial products use PCBs for their various benefits including being *faster to assemble*. Should we ever wish to mass produce, (2) is essentially a necessity. However, when prototyping, it is an *additional step* and *not flexible* on the spot, unlike (1) so there is a place for both in the design process.