Unifying Command Selection Mechanisms on Mobile Devices with Soft Keyboard Shortcuts (SoftCuts)

Submitted by

Katherine FENNEDY

Thesis Advisors

Dr. Hyowon LEE and Dr. Simon PERRAULT

Information Systems Technology and Design Pillar

A thesis submitted to the Singapore University of Technology and Design in fulfilment of the requirement for the degree of Doctor of Philosophy

2021
PhD Thesis Examination Committee

Main Advisor: Dr. Simon PERRAULT
Co-advisor(s): Dr. Hyowon LEE
TEC Chair: Prof. Ngai-Man CHEUNG
Internal TEC member 1: Dr. Dorien HERREMAN
Internal TEC member 2: Dr. LIM Kwan Hui
External TEC member 1: Prof. Andy COCKBURN
Abstract

Information Systems Technology and Design Pillar

Doctor of Philosophy

Unifying Command Selection Mechanisms on Mobile Devices with Soft Keyboard Shortcuts (SoftCuts)

by Katherine FENNEDY

Despite their mainstream availability, mobile devices do not support a unified yet efficient command selection mechanism available on every platform and application. We advocate that hotkeys, conventionally used as a shortcut mechanism on desktop computers, could be generalised as a command selection mechanism for touch-based mobile devices, even for scenarios where the soft keyboard is not immediately visible. In this thesis, we capitalise on the consistent keyboard layout to offer a unified command selection mechanism suitable for both novice and expert users. We introduce this approach under the name SoftCuts which is an abbreviation for soft keyboard shortcuts. We conducted seven studies across five different projects to investigate the qualitative and quantitative benefits of SoftCuts. Our results show that SoftCuts could leverage users’ prior knowledge of hotkeys and offer a competitive advantage in selection performance and learnability, which are essential elements in maximising usability for a user interface.
Publications

In reverse chronological order:

   https://doi.org/10.1109/TVCG.2021.3101854

   https://doi.org/10.1145/3476510

   https://doi.org/10.1145/3447526.3472038

   https://doi.org/10.1145/3379503.3403552

   https://doi.org/10.1145/3335595.3335645

   https://doi.org/10.1145/3335595.3335598

   https://doi.org/10.1145/3328243.3328249

   https://doi.org/10.14236/ewic/HCI2018.9
Acknowledgements

This thesis would not have been possible without the love from many individuals and groups. In many ways, they are my support system, from guiding me when I had to deal with setbacks or feeling lost, to celebrating my every milestones, be it big or small. In particular, I would like to thank:

- **My PhD advisors: Hyowon Lee and Simon Perrault.** They both contributed significant time and effort in nurturing me into the better and more confident researcher I am today. I will never forget when Hyowon planted the idea of pursuing a doctoral degree in my head despite my limited research understanding back then. His trust in my potential capabilities moved me to grab this once-in-a-lifetime opportunity and give my very best. Simon's genuine mentorship further accelerated my research developments and sharpened my skills. He directed my interests and ideas into meaningful projects by connecting me with experts in the field. Even if I could rewrite history, I would still choose them both as my advisors to guide me in this uncertain yet rewarding journey of research.

- **My SHUAILab mentors and friends: Sylvain Malacria, Kenny Choo, Atima Tharatipyakul, Nguyen Thi Ngoc, Pavithren V S Pakianathan, Nurhadi Ahmad, Gao Jie, Gionnieve Lim, and Angad Srivastava.** They offered alternative research perspectives by sharing feedback and giving me frequent opportunities to rehearse my presentations. I wish them great success with their respective research journey.

- **My EXII mentors and friends at University of Waterloo: Daniel Vogel, Quentin Roy, Hemant Surale, Jeremy Hartmann, Matthew Lakier, and Nikhita Joshi.** They taught me how to conduct research with a different style and what it means to lead with kindness. I am thankful to have spent months of research with them during winter before the Covid-19 pandemic recalled me back to Singapore.

- **My SUTD PhD sisters: Koh En Yan, Marie Siew, Verina Cristie, Sophia Chan, Denise Lee, Soh Mei Yu, Penny Chong, Thilini Cooray, Zeynep Duygu Tekler, and Teo Ting Yu.** They showered me with love through the occasional food outings, study-together sessions, or heart-to-heart talks.

- **My SUTD brothers: Ahnaf Siddiqi, Dhrubajit Chowdhury, Luccan Ryanata, Alvee Ahmed, Ismam Al Hoque, and Adeeb Hossain.** They showed me that my life’s worth is not limited to my research contributions through frequent food-for-thought discussions and board game sessions.

- **My PADI brothers and sisters: Anthony Setiawan, Yehezkiel Wiliardy Manik, Stevanus Satria, Hendrik Teguh, Michelle Gouw, Michael Kanta, Jessica Davinia, and many more.** They were my home away from home.

- **My special individuals: Gede Agus Aswamada and Ragini Gurumurthy.** They made me feel safe whenever I shared vulnerable moments of anxiety and despair. With their help, I learn to minimise my temptation to overthink, embrace the unknown, and trust my gut instinct.
• My first and forever family: Nelly Kasmir, Bill Fennedy, and Chelsea Fennedy. They are my constant source of motivation and the reason why I do not give up when faced with roadblocks. I am forever indebted to them for believing in me regardless of how crazy the idea to pursue a PhD seems to be initially.

Thank you everyone for being a part of this special chapter of my life.
# Contents

PhD Thesis Examination Committee .................................................. i
Abstract .......................................................................................... ii
Publications ................................................................................... iii
Acknowledgements ......................................................................... iv

1 Introduction .................................................................................. 1
  1.1 Research Motivations ............................................................... 1
  1.2 Research Goals ........................................................................ 4
    1.2.1 Term Definitions .............................................................. 4
    1.2.2 Thesis Statement .............................................................. 5
    1.2.3 Research Questions .......................................................... 7
  1.3 Research Contributions ............................................................ 9
    1.3.1 Design Contribution ......................................................... 9
    1.3.2 Empirical Contribution ..................................................... 10

2 Related Work ............................................................................... 13
  2.1 Mode ....................................................................................... 13
    2.1.1 Definition ......................................................................... 13
    2.1.2 Significance ...................................................................... 15
    2.1.3 Mode-Switching Behaviours ............................................. 16
    2.1.4 Relevance to SoftCuts ....................................................... 19
  2.2 Command Selection ................................................................. 19
    2.2.1 With menus ...................................................................... 20
      Using keyboard layouts .......................................................... 23
    2.2.2 With gestures ................................................................... 24
      Using keyboard layouts .......................................................... 26
    2.2.3 With hotkeys .................................................................... 28
  2.3 Skill Development .................................................................... 31
    2.3.1 Skill Acquisition .............................................................. 32
    2.3.2 Skill Retention ................................................................. 35
    2.3.3 Skill Transfer ................................................................... 36
    2.3.4 Relevance to SoftCuts ....................................................... 37
  2.4 Why SoftCuts? ......................................................................... 38
### 3 Optimising Moded Interaction Design for Soft Keyboard

- **3.1 Motivation** ................................................. 39
- **3.2 Approach** .................................................. 40
  - 3.2.1 Layer 1: Between Languages 41
  - 3.2.2 Layer 2: Within a Language 43
    - The Inception of SoftCuts 47
- **3.3 Quantitative Evaluation** .......................... 48
  - 3.3.1 Layer 1 Comparison: Between languages 48
  - 3.3.2 Layer 2 Comparison: Within a Language 50
- **3.4 Discussion** .............................................. 50
- **3.5 Conclusion** ............................................... 51

### 4 Maximising Discoverability of SoftCuts

- **4.1 Motivation** .................................................. 53
- **4.2 Study 1: Visual Representation of Commands** 53
  - 4.2.1 Design Variants 54
    - D1: Letter only. 54
    - D2: Letter and name. 54
    - D3: Icon and name. 55
  - 4.2.2 Procedure 55
  - 4.2.3 Task 56
    - Note-taking App (Keyboard Task) 56
    - Web-browsing App (Keyboardless Task) 57
  - 4.2.4 Participants and Apparatus 57
  - 4.2.5 Design 58
  - 4.2.6 Results 59
    - Notes (Keyboard Scenario) 59
    - Browser (Keyboardless Scenario) 59
  - 4.2.7 Discussion 59
  - 4.2.8 Limitations 62
- **4.3 Study 2: Familiarity and Saliency of Modifier Keys** 62
  - 4.3.1 Missions and Tasks 63
  - 4.3.2 Procedure 64
  - 4.3.3 Measuring Discoverability 64
  - 4.3.4 Participants and Apparatus 65
  - 4.3.5 Design 66
  - 4.3.6 Results 66
  - 4.3.7 Discussion 70
  - 4.3.8 Limitations 71
- **4.4 Conclusion** ............................................... 71

### 5 Evaluating Input Methods for SoftCuts

- **5.1 Motivation** .................................................. 73
- **5.2 Study 1: Performance of Input Methods** 74
  - 5.2.1 Participants and Apparatus 74
  - 5.2.2 Procedure 74
  - 5.2.3 Tasks 75
List of Figures

1.1 (a) A physical keyboard by Keychron (Keychron, 2021). (b) Hotkey labels are displayed to the right of their corresponding commands in a menu of Microsoft Word application. ................................................. 2
1.2 Soft keyboards displayed in a Notes app of (a) Apple and (b) Samsung phones while in portrait mode. .................................................. 3
1.3 Soft keyboards displayed in a Notes app of (a) Apple and (b) Samsung tablets while in landscape mode. .................................................. 3
1.4 Existing Implementations of Soft Keyboard Hotkeys on (a) Samsung Galaxy S, with the hotkeys highlighted in blue once the user presses the Ctrl key, (b) Microsoft Surface, with the name of the commands associated with hotkeys displayed after the user taps on the Ctrl key. ................. 4
1.5 Core chapters of the thesis and their relationships in connecting Soft-Cuts’ research exploration journey from Chapter 3 to 7. .................. 6

2.1 An interaction model of a system with one input and three distinct modes, illustrating that mode ‘2’ is responsible for the current blue output. Red and yellow output are associated with mode ‘1’ and ‘3’ respectively. . . . 14
2.2 Examples of non-computer devices that adopt mode, as listed by Jeff Johnson (1990). .............................................................. 14
2.3 Examples of computer devices that adopt mode: (a) sketching app by Wacom (2020) and (b) a tabletop setup demonstrating interaction by TapSense (Harrison, Schwarz, and Hudson, 2011). ................................. 15
2.4 Organisation of commands in Microsoft Word desktop application. . . 20
2.5 Spatially stable command selection interfaces: (a) CommandMaps from J. Scarr, Cockburn, Gutwin, and Bunt (2012), (b) StencilMaps and (c) EphemeralMaps from J. Scarr, Gutwin, et al. (2015). .......................... 21
2.6 Food-inspired user interface (UI) elements for menu navigation design on mobile devices. ............................................................. 21
2.7 Hierarchical navigation of commands in Microsoft Word tablet application. Tapping on the meatballs menu (a) will reveal more commands like (b), which act as sub-menus to more commands in (c). .................. 22
2.8 Hierarchical navigation of commands in Microsoft Word smartphone application. Selecting a text element will reveal a keyboard and a scrolling toolbar (a), which features a subset of commands. The full set (c) can be accessed by either tapping on the meatballs menu in (a) or the text-editing icon (b). However, tapping the meatballs menu (d) will reveal a different set of commands, like (e) and (f), related to the whole document instead of a specific text element. .................. 22
2.9 Grid-based and Bezel-based menu layouts: (a) FastTap (from Gutwin, Cockburn, J. Scarr, et al. (2014)), (b) HandMark Menus (from Uddin, Gutwin, and Lafreniere (2016)), and (c) hidden toolbar (from Schramm, Gutwin, and Cockburn (2016)).

2.10 Techniques using keyboard layouts to organise command items to be selected or document items to be retrieved: (a) Touch-Display Keyboards (from Block, Gellersen, and Villar (2010)), (b) Hotkey Palette (from Aceituno and Roussel (2014)), and (c) Optimus Popularis keyboard (from Studio (2011)).

2.11 Techniques using stroke gestures: (a) Marking Menus (from Gordon Kurtenbach and B. Buxton (1994)), (b) Crib-Sheet (from G. Kurtenbach, T. P. Moran, and W. Buxton (1994)), (c) OctoPocus (from Bau and Mackay (2008)), and (d) Augmented Letter (from Roy et al. (2013)).

2.12 Techniques using hand gestures: (a) ShadowGuides (from Freeman et al. (2009)), (b) MelodicTap (from Heo, Jung, and G. Lee (2016)), (c) Arpège (from Ghomi et al. (2013)), and (d) Gesture Play (from Bragdon, Uguray, et al. (2010)).

2.13 Techniques using keyboard layouts to frame stroke gestures: (a) Command Strokes (from P. O. Kristensson and Zhai (2007)), (b) HotStrokes (from Cui et al. (2019)), (c) CommandBoard (from Alvina, Griggio, et al. (2017)), and (d) GestKeyboard (from Zhang and Y. Li (2014)).

2.14 Strategies to boost hotkey usage: (a) Skillometers (from Malacria, J. Scarr, et al. (2013)), (b) ExposeHK (from Malacria, Bailly, et al. (2013)), (c) IconHK (from Giannisakis et al. (2017)), and (d) KeyMap (from Lewis, d'Eon, et al. (2020)).

2.15 Techniques to extend input vocabulary of hotkeys: (a) Finger-Aware Shortcuts (from Zheng and Vogel (2016)), (b) FingerArc and FingerChord (from Zheng, Lewis, et al. (2018)), (c) Métamorphe (from Bailly, Pietrzak, et al. (2013)), and (d) by rotating arm and wrist (from Buschek, Roppelt, and Alt (2018)).

2.16 Performance curve from Cockburn, Gutwin, et al. (2014), adapted from J. Scarr, Cockburn, Gutwin, and Quinn (2011).

2.17 Guided interfaces that support discovery and user learning: (a) GestureBar (from Bragdon, Zeleznik, et al. (2009)) for gestures and (b) HotKeyCoach (from Krisler and Alterman (2008)) for hotkeys.

2.18 TouchTools designed virtual tools (top) to replicate the real-world grasps (bottom). Image is obtained from Harrison, Xiao, et al. (2014).

2.19 Illustrations of user performance transfer: (a) between physical and digital interaction (from Bérard and Rochet-Capellan (2015)) and (b) between proactive and retroactive type (from Raissi et al. (2020)).

3.1 Code-switching between 3 languages: (a) English, (b) Emoji and (c) Korean.

3.2 Existing model of mode-switching between languages.

3.3 Instead of looping through intermediate modes, one can hold (left) on globe icon and tap (right) on a desired mode.
3.4 Proposed UI design for mode-switching between languages. Changes were illustrated using red boxes. ................................. 42
3.5 Proposed model of mode-switching between languages. .................. 43
3.6 Existing UI design for mode-switching within Latin/QWERTY mode. 44
3.7 Existing model of mode-switching interaction within Latin/QWERTY mode. ......................................................... 45
3.8 Proposed model of mode-switching interaction within Latin/QWERTY mode. ......................................................... 46
3.9 Proposed UI design for mode-switching within Latin/QWERTY mode. Changes were illustrated using red boxes. .............. 47
3.10 Evaluation of strategies switching between languages by bilinguals. 48
3.11 Evaluation of strategies switching modes within a language to type a password or hashtag: #CH4UXID2019. The password is inspired by the conference name ‘CHIuXiD’ where we presented this chapter of this thesis. 49
3.12 Evaluation of strategies switching modes within a language to type a mathematical expression. .................................... 49
3.13 Evaluation of strategies switching modes within a language to type in a programming syntax. ............................................. 50
4.1 Design variations for SoftCuts. (a) We modified the existing iOS keyboard by adding two command keys at the bottom row. (b) Design 1: Available shortcut keys retain their default appearance, while non-shortcut keys are greyed and inactive. (b) Design 2: In addition to Design 1, the name of each command is printed at the bottom of respective keys. Design 3: Similar to Design 2, but substituting the key labels with icons instead. ............................................................... 54
4.2 A demonstration of tasks in a note-taking app that may be done using existing mechanisms or SoftCuts’ invocations to complete. Participants had to paste text, (a) select all the pasted text, (b) bold the selected text, (c) undo the bold formatting then select all the text again, and finally (d) colour the selected text red. Step (c) and (d) could only be achieved through SoftCuts. ............................................................... 56
4.3 A demonstration of the search task in web-browsing app that can only be completed using SoftCuts. Participants had to (a) press the command key at the bottom left, then (b) the soft keyboard would appear with active keys that correspond to available keyboard shortcuts, and when they would press the ⋆ key, (c) the search bar would appear above the keyboard with a preset text and the page will animate a scrolling down effect until the text has been identified. ............................................................... 58
4.4 Subjective rating for each DESIGN of NOTES app (1: Strongly Disagree, 7: Strongly Agree). ..................................................... 59
4.5 Subjective rating for each DESIGN of BROWSER app (1: Strongly Disagree, 7: Strongly Agree). ..................................................... 60
4.6 The bottom section of starting screen while varying both FAMILIARITY and SALIENCY: (a) Keyboard (KB) + Command as modifier, (b) KB + Custom modifier, (c) Keyboardless (noKB) + Command, and (d) noKB + Custom. ..................................................... 63
4.7 A desktop version of the mobile prototype being deployed to each crowdworker.

4.8 *Spontaneous discovery rate* of SoftCuts (a) across all conditions, (b) aggregated by FAMILIARITY (i.e. type of modifier key used), and (c) aggregated by SALIENCY (i.e. presence of keyboard).

4.9 *Enforced discovery rate* of SoftCuts (a) across all conditions, (b) aggregated by FAMILIARITY (i.e. type of modifier key used), and (c) aggregated by SALIENCY (i.e. presence of keyboard).

4.10 *Overall discovery rate* of SoftCuts (a) across all conditions, (b) aggregated by FAMILIARITY (i.e. type of modifier key used), and (c) aggregated by SALIENCY (i.e. presence of keyboard).

4.11 *Spontaneous discovery rate* of SoftCuts across (a) frequency of hotkey usage, (b) quantity of hotkey known/used, and (c) years of experience using hotkeys.

4.12 *Enforced discovery rate* of SoftCuts across (a) frequency of hotkey usage, (b) quantity of hotkey known/used, and (c) years of experience using hotkeys.

4.13 *Overall discovery rate* of SoftCuts across (a) frequency of hotkey usage, (b) quantity of hotkey known/used, and (c) years of experience using hotkeys.

4.14 Subjective rating for each CONDITION (1: Strongly Disagree, 7: Strongly Agree).

5.1 State diagrams for each input method, showing that all three can be used concurrently on a system. Note: UM allows user to perform multiple command selections by tapping multiple hotkeys sequentially.

5.2 Example of a trial on smartphone. (a) The stimulus is highlighted in blue at the beginning of the trial to minimise visual search (b) the user pressed on one of the modifier keys, (c) upon correct selection, the key would turn green (red if incorrect).

5.3 (a) Selection time and (b) accuracy performance for the one-handed conditions. Error bars show 95% confidence intervals. Accuracy ordinate axis starts at 75%.

5.4 Selection times for each INPUT METHOD on both phone and tablet while using 2 hands. Error bars show 95% confidence intervals.

5.5 Selection times for each ORIENTATION on both phone and tablet while using 2 hands. Error bars show 95% confidence intervals.

5.6 Accuracy performance for individual input method on both phone and tablet for the two-handed conditions. Error bars show 95% confidence intervals. Ordinate axis starts at 75%.

5.7 Subjective rating for each LAYOUT grouped by DEVICE: (top) PHONE and (bottom) TABLET (1: Strongly Disagree, 5: Strongly Agree).

5.8 Experimental setup for the treadmill conditions.

5.9 Layout for our second experiment, each shortcut is a country name with its flag. Left: layout on a phone in portrait mode, Right: layout on the tablet in landscape mode.
5.10 Distribution of each Input Method (UM, Once, Swipe) across 4 conditions between device, handedness, orientation while sitting, standing and walking. (a) is derived from actual usage during the study while (b) is the declared preference at the end of each condition.

6.1 Example of a trial in the experiment. (a) The participant was first presented with the stimulus "Australia", (b) they clicked on one of the left/right modifier keys at the bottom of the screen, then (c) they accurately selected "Australia" and received feedback. The abstract command set is shown on the (b) and (c) panels.

6.2 Layout of our realistic command set, taken from the Windows version of Microsoft Word.

6.3 (a) Time by BLOCK for each MAPPING. (b) Accuracy by BLOCK for each MAPPING. Error bars are 95% confidence.

6.4 (a) Retention rate by PHASE (at the end of the experiment (0D), after 1 day, 3 days, and 7 days) for each MAPPING. “Correct” rates indicate that the participant recalled the right key, while “Correct+Adjacent” also includes recalls on one of the adjacent keys. Error bars are 95% confidence. (b) Participants tended to misplace shortcuts from the target location (e.g. in green) to one of the keys adjacent to the target (e.g. in orange).

6.5 Subjective rating for each MAPPING using a 5-point Likert scale (1: Strongly Disagree, 5: Strongly Agree).

7.1 Demonstration of how a selection trial is completed by each participant on the phone prototype for both spatial layouts: keyboard (a to c) and grid (d to f). (a) and (d) is when the target stimulus is first presented while the modifier key(s) is/are placed near the bottom of the screen, then (b) and (e) is when the participant uses one of their finger to hold onto a modifier key while using another finger to press the target command. (c) and (f) show the visual feedback when the selection is completed correctly.

7.2 Layouts displaying (a) activity, (b) animal, (c) flag, and (d) food command mappings on a tablet prototype. (a) and (b) use a keyboard layout while (c) and (d) use a grid layout (similar to FastTap).

7.3 (a) Time by BLOCK for each LAYOUT and DEVICE. (b) Time by DEVICE for each LAYOUT while excluding blocks that demonstrated learning effect. Error bars are 95% confidence.

7.4 (a) Accuracy by BLOCK for each LAYOUT and DEVICE. (b) Accuracy by DEVICE for each LAYOUT. Error bars are 95% confidence.

7.5 Subjective rating for each LAYOUT grouped by DEVICE: (a) PHONE and (b) TABLET (1: Strongly Disagree, 5: Strongly Agree).

8.1 (a) Example of SoftCuts on an Android soft keyboard with a ctrl modifier key. (b) Tapping ctrl once displays the first level of shortcuts (e.g. ctrl+s for save), and substitutes #1 with alt. (c) Tapping on alt displays additional shortcuts that are usually triggered with both ctrl and alt modifier keys.
8.2 Worldwide market share of desktop vs. smartphone vs. tablet. Data source: https://gs.statcounter.com/ ........................................ 108

8.3 Strategies using (a) virtual and (b-d) physical keyboard to support interaction in VR. (a) is from Speicher et al. (2018), (b) is from Grubert, Witzani, et al. (2018), (c) is from Knierim et al. (2018), and (d) is from D. Schneider et al. (2019). ...................................................... 109

8.4 Proposed design for SoftCuts in AR or MR environment. ...................... 110

8.5 Proposed design for SoftCuts in VR environment. ............................... 111

9.1 Web browsing scenario of soft keyboard shortcuts using Once method. (a) a semi-transparent modifier key is available in the browser, (b) when user taps it, all available shortcuts will be presented to facilitate browsing, (c) user then taps on [f] (for ‘find’ command) and hence, (d) every occurrence of the “HCI” word (as typed by the user) is highlighted in yellow. ................................................................. 114

E.1 Keyboard Layout Reference. ...................................................... 126
List of Tables

4.1 Design space for the visual representation, using Letter (key name), Name (of command) and Icons. .............................................. 54

5.1 Experimental Conditions for Study 1. Greyed-out conditions were excluded. ......................................................... 75
List of Abbreviations

D1    Design Variant #1: keys are represented using only letters
D2    Design Variant #2: keys are represented using command names and letters
D3    Design Variant #3: keys are represented using command names and icons
GUI   Graphical User Interface
HCI   Human-Computer Interaction
MM    Marking Menu
OS    Operating System
PC    Personal Computer
SoftCuts Soft keyboard ShortCuts
UM    User-Maintained
UI    User Interface
VR    Virtual Reality
WIMP  Windows, Icons, Menus, and Pointers
A tribute to my grandparents
Kwan Thien Fu & Jio Siu Kim
and my mother Nelly Kasmir

whose sacrifices have made it possible for me to pursue a
high-quality education overseas. Thank you Agong, Ama, and
Mama for sharing the joy of lifelong learning and inspiring me to
always give my best in everything I do.

With this thesis, I can finally answer my Ama’s favourite
recurring question, "when are you going to graduate?".

Rest in peace Mama.
Chapter 1

Introduction

1.1 Research Motivations

Imagine you are on your phone right now, typing on a Notes app, and you accidentally deleted a large portion of your texts. Do you know that you could retract such a common user error? If yes, do you know how? Most people would first intuitively search for the "Undo" icon or text label. Unfortunately, only a handful of apps provide access to undo command directly through visual cues because mobile devices like our smartphones have limited screen real estate, so certain features will be prioritised and made visually salient. For iOS devices, users could use either shake-to-undo\(^1\) or three-fingers-swipe\(^2\) gestures to trigger the undo or redo command regardless of the app being used. For Android devices, users could install third-party apps like Inputting Plus\(^3\) to trigger "Undo", "Redo", or "Find & Replace" commands, especially when their access is not made readily available. Most users would not have discovered these features (Alvina, Bunt, et al., 2020) if it is not through word-of-mouth, or curiosity that drives them to explore the world wide web and stumble upon articles that promote "hidden tricks every Android & iOS user should know"\(^4\).

Beyond poor discoverability, the bigger problem we have to acknowledge is that today's touch-based devices have yet to offer a unified command selection mechanism that is reliably fast and available on every platform and application. Tapping is one example of command selection mechanism that is ubiquitously adopted by multitouch phones and tablets. However, the downside of its simplicity is that it is not scalable for large command sets which may require tapping through menu hierarchies, and thus slowing down the selection. Another point is that one might know how to trigger a command on an iOS device, but it does not necessarily translate to an awareness that the same command automatically exists on an Android device. Furthermore, even when it exists on different Operating Systems (OS), the input method required may have been implemented differently (shake vs. swipe vs. menus). This inconsistent mapping between user input and command output goes against the usability guidelines established in the Human-Computer Interaction (HCI) field for decades (Nielsen

---

\(^2\)https://developer.apple.com/design/human-interface-guidelines/ios/user-interaction/gestures/
\(^3\)https://play.google.com/store/apps/details?id=com.catchingnow.undo
\(^4\)http://www.ecoti.in/DwTDMZ48
Chapter 1. Introduction

Figure 1.1: (a) A physical keyboard by Keychron (Keychron, 2021). (b) Hotkey labels are displayed to the right of their corresponding commands in a menu of Microsoft Word application.

and Molich, 1990; Shneiderman, 1997). Users are forced to constantly learn new techniques and re-optimise their performance with every new system that does not promote inter-usability (Denis and Karsenty, 2003). It is especially relevant in today’s context where the ubiquitous nature of technology has allowed users the ability to alternate between multiple devices or even use them simultaneously to complete a task. Therefore, it is now more important than ever to support users with a consistent yet versatile mechanism that could maximise users’ productivity instead of struggling with the nuts and bolts of an interface.

Despite the difference in device sizes and interface design guidelines across existing OS, one common element is shared between any smartphone and any tablet but remains surprisingly untapped, and that is a soft keyboard. Soft (or digital/on-screen) keyboards are inspired and based on physical (or mechanical) keyboards for computers (see Fig. 1.1a). Layouts are usually similar, with slight alterations due to space constraints (see Fig. 1.2 and 1.3).

The primary role of physical keyboards is to enable text-entry tasks on desktop computers. Their secondary role is to provide experts with shortcuts/accelerators through the chording of multiple physical keys. These keyboard shortcuts (also known as hotkeys) are relatively established in terms of efficient performance (Card, Thomas P. Moran, and Newell, 1980) and consistent mapping across OS, as compared to a menu-based or gesture-based activation. Examples of frequently used hotkeys include $\text{⌘} + \text{C}$ for “Copy” and $\text{⌘} + \text{V}$ for “Paste” command (see Fig. 1.1b). Soft keyboards are only made available on touch-based devices when users need to enter text (e.g. Notes application). This might be one reason why hotkeys never seem to be considered as a viable standard to select commands on mobile devices, despite user familiarity while using desktop computers. Therefore, we are motivated to study how the under-exploited hotkeys functionality can be extended from physical keyboards to soft keyboards.

Two reasons primarily drive our motivations. First, the keyboard layout provides a familiar canvas to maintain spatial consistency across applications for a given command, as keyboard layouts are mostly consistent across devices and even compared
Chapter 1. Introduction

Figure 1.2: Soft keyboards displayed in a Notes app of (a) Apple and (b) Samsung phones while in portrait mode.

Figure 1.3: Soft keyboards displayed in a Notes app of (a) Apple and (b) Samsung tablets while in landscape mode.
to physical keyboards. Second, soft keyboard shortcuts (SoftCuts) can become a multi-device command selection mechanism that allows advanced users to reuse pre-existing knowledge from using desktop computers to mobile devices, or vice versa.

Although attempts to instantiate SoftCuts can be found on some commercial touch-based devices like the latest Microsoft Surface (PandaSage1221, 2013) and Samsung Galaxy (Samsung, n.d.) tablets (see Fig. 1.4), they do not leverage the dynamic nature of on-screen keyboard layout that could be changed at will. For instance, the commands associated with each key may not be expressively shown to the user, who has to either guess them or discover them through repeated tries. Furthermore, it is unclear why only a subset of all the commands was rendered as available and only during limited scenarios where the keyboard has to be displayed. Hence, there is certainly ample room for opportunities to correct existing designs and add complementary features to augment the potential of SoftCuts. Our works represent the first formal and thorough investigation that reveals why and how SoftCuts could be generalised as a viable command selection mechanism for diverse applications on mobile devices.

1.2 Research Goals

Before we elaborate on the overarching goal and specific research questions, we first define the terms used throughout this thesis.

1.2.1 Term Definitions

1. Mobile Devices - We use the term "mobile devices" to refer to any handheld computers that support interactions in diverse mobility conditions that include sitting, standing, and walking. Our work focuses on touch-based devices like smartphones and tablets. We exclude smartwatch from our studies as most of them rely on smartphones to run the majority of the function, and instead can be considered a smartphone accessory.
2. **App** - We use the term "app" to refer to any software application that can be installed on any device. Examples of an app include Evernote\(^5\) for note-taking, Adobe Lightroom\(^6\) for photo-editing, and Sculptura\(^7\) for 3D-modelling.

3. **Command Selection** - We use the term "command selection" to refer to tasks where a user activates one app functionality from a bounded set of functions. Examples of a command include "Copy", "Paste", "Undo", "Crop", "Extrude", and "Intersect".

4. **Gesture** - We use the term "gesture" to refer to a sequence of human actions executed automatically once set in motion (Raskin, 2000). Examples of a "gesture" include pressing, tapping, hovering, holding, swiping, or strokes.

5. **Input Method** - We use the term "input method" to refer to interaction techniques or gestures across different modalities (e.g. pen-based, mouse-based, touch-based) to support users in accomplishing their desired tasks.

6. **Novice** - We use the term "novices" to refer to a user who either has never used the system or has rarely used it. Hence, they need guidance and time to learn, remember, and practice how to execute the input method best.

7. **Expert** - We use the term "expert" to refer to a user who has used the system so often that they no longer need any guidance and can rely on muscle memory to execute the input method.

### 1.2.2 Thesis Statement

We hypothesise that a universally consistent keyboard layout can be capitalised to offer a unified and efficient command selection mechanism on mobile devices, suitable for both novice and expert users. We will dissect this hypothesis of ours through five core projects into a series of more specific research questions to delve deeper. First, we explore how the concept of *mode* can be exploited to redesign soft keyboards in the context of text-entry scenarios. Second, we extend our focus to command selection scenarios by investigating how *visual design elements* of SoftCuts affect its discoverability for novices. Third, we investigate how different *input methods* of SoftCuts affect the performance and usage of SoftCuts across diverse mobility conditions. Fourth, we incorporate the ideal input method and study how the *command mappings* affect novices’ long-term retention and experts’ ability to leverage prior knowledge of SoftCuts. Last but not least, we compare the performance of command selections between SoftCuts’ keyboard-based *layout* and an alternative grid-based layout.

The core contribution of this thesis is illustrated in Fig 1.5, where we summarise each chapter’s problem statement, research methodology, and key contribution. We start by building the foundation through understanding the soft keyboard’s typing functionality (Chapter 3) before delving deeper into its keyboard shortcuts functionality. The main contribution of this thesis revolves around demonstrating the why

---

\(^5\)[https://evernote.com/]
\(^6\)[https://lightroom.adobe.com/]
\(^7\)[https://sculptura.app/]
Chapter 1. Introduction

Chapter 3
How do mode-switching analysis improve soft keyboard typing efficiency?
We analyse existing model of interaction and reveal how proposed enhancements could offer a more optimised interface designs.
Recommended strategies to address complex mode hierarchy and incomplete mode transitions by adopting user-maintained (UM) interaction techniques. New keyboard designs introduce novel opportunities for command selection shortcuts, which we call as SoftCuts.

Chapter 4
What are the recommended visual design strategies to maximise discoverability of SoftCuts?
We conduct usability evaluations between designs varying representations of on-screen keys, saliency and familiarity of modifier key.
Keys are best represented with both names and icons of commands, but substituting icons with letters is a good alternative for expert users. It is easier to discover SoftCuts when the user is familiar with using hotkeys on desktop. Familiarity and Salience of modifier key does guarantee higher discoverability for SoftCuts.

Chapter 5
What is a robust input method for SoftCuts?
We compare speed, accuracy, and adoption of each input method across various device configurations and mobility conditions.
Using two sequential taps to trigger SoftCuts is most robust, but swiping and UM are suitable for phone and tablet interaction respectively.

Chapter 6
How do command mappings affect user learning for SoftCuts?
We compare speed, accuracy, and retention rate between abstract and realistic mappings.
Both mappings allow participants to leverage prior knowledge and remember command locations even after a week, with minimal difference in selection performance.

Chapter 7
How do spatial layouts affect selection performance for SoftCuts?
We compare speed, accuracy, and subjective preferences between a keyboard and a grid layout on a phone and a tablet.
Despite a faster selection with a grid layout, its accuracy is comparable with SoftCuts' keyboard layout which offers a greater spatial stability occupying a much reduced screen real estate.

Figure 1.5: Core chapters of the thesis and their relationships in connecting SoftCuts' research exploration journey from Chapter 3 to 7.
and how SoftCuts designs could offer a more unified command selection mechanism (Chapter 4-7), whose performance is robust across applications and adaptive to users’ level of expertise.

1.2.3 Research Questions

Optimising Modeled Interaction Design for Soft Keyboard

The first project is inspired by a previous mode analysis on non-computer devices conducted three decades ago (Jeff Johnson, 1990). We re-orient mode and its mechanism in today’s context of modern computing devices, especially with the ubiquitous interaction modalities such as multitouch.

In this project, we address the following research questions:

1. What are the inefficiencies identified by running mode analysis on multitouch typing?
2. What strategies can be adopted to address the inefficiencies?
3. How do proposed designs compare to the existing approach of mode integration?

Maximising Discoverability of SoftCuts

The second project is our first direct investigation on SoftCuts: the keyboard shortcut functionality of soft keyboards for multitouch command selections. We focus on optimising the design of SoftCuts’ visual elements across two studies. This is because commercial attempts by Microsoft (PandaSage1221, 2013) and Samsung (Samsung, n.d.) to adopt SoftCuts do not leverage the dynamic nature of soft keyboards’ layout that could be changed at will. Soft keyboards could make the discovery and learning of hotkeys even more effortless than on desktop systems.

In this project, we address the following research questions:

1. Which key representation is suitable for SoftCuts?
2. How does saliency of modifier key affect the discoverability of SoftCuts?
3. How does familiarity of modifier key label affect the discoverability of SoftCuts?

Evaluating Input Methods for SoftCuts

The third project compares input methods for SoftCuts in terms of performance and usage across device configurations and mobility conditions. Commercial attempts by Microsoft (PandaSage1221, 2013), Samsung (Samsung, n.d.), and Swype app (Nuance, 2017) adopt input methods that are all different from one another. However, there has not been any investigation to compare these input methods and find out which one(s) may be better for users in terms of objective and subjective performance. It is crucial to standardise the command selection mechanism for SoftCuts such that users do not
Chapter 1. Introduction

have to relearn techniques as they switch between devices. It is also somewhat unclear if activities like standing and walking could affect the performance and usage of these input methods. This is especially relevant today to investigate because the ubiquitous nature of our mobile devices has made it possible to complete tasks whenever and wherever one demands.

In this project, we address the following research questions:

1. How do different device configurations affect the performance of input methods?
2. How do different mobility scenarios affect the performance of input methods?
3. How do different mobility scenarios affect the usage of input methods?
4. How do different device configurations affect user preference of input methods?

Leveraging Prior Knowledge and Sustaining Retention

The fourth project is an evaluation of selection performance and retention between command mappings. The layout of the keyboard provides an efficient strategy to search a desired key/letter visually. However, it is the mapping between each key and a command that will ultimately influence the ceiling performance of user selections. Novices, being unfamiliar with a given mapping, will need time and effort to build retention for the new knowledge. On the other hand, experts can recall the learned mapping by memory, and we hypothesise that they could transfer this advantage to a new system that adopts the same layout and mapping. In our case, users familiar with specific keyboard shortcuts on their desktop computer may benefit from reusing similar soft keyboard shortcuts on their mobile devices. Therefore, we need to study this unexplored extent to which command mappings affect the learning progress for both novices and experts.

In this project, we address the following research questions:

1. How do command mappings affect experts’ ability to leverage prior knowledge?
2. How do command mappings affect novices’ ability to retain knowledge?

Comparing Performance between Keyboard and Grid Layout

The fifth project is an assessment of how SoftCuts’ keyboard layout stand against an alternative grid layout. Spatially consistent (J. Scarr, Cockburn, Gutwin, and Malacria, 2013) interfaces are robust to typical view transformations because they maintain stable locations of the objects relative to a particular frame of reference. Both keyboard and grid layouts are spatially consistent. Since the latter has been adopted by Fast-Tap (Gutwin, Cockburn, J. Scarr, et al., 2014) and has shown to be fast and accurate on many touch-based devices (Lafreniere, Gutwin, Cockburn, and Grossman, 2016; Lafreniere, Gutwin, and Cockburn, 2017; Goguen, Malacria, Cockburn, et al., 2019a), it offers the closest baseline we can use to quantify SoftCuts’ impact of relying on a keyboard
In this project, we address the following research questions:

1. How do spatial layouts affect selection speed?
2. How do spatial layouts affect selection accuracy?

### 1.3 Research Contributions

#### 1.3.1 Design Contribution

**Optimising Moded Interaction Design for Soft Keyboard**

In Chapter 3, we describe a study to analyse typing interaction on a soft keyboard and re-frame its mode perspective. We used StateChart (Harel, 1987) to model the mode behaviour in two layers: between languages and within a language. This helps us to better understand the inefficiencies in existing transitions and the relationship between modes. We then proposed improvements in terms of the interaction model and, correspondingly, the user interface design of the system. To validate them, we theoretically approximated the required time or effort by counting the number of taps needed by the different model to type the same given string of characters.

The study was published at CHIuXiD 2019 (Fennedy and H. Lee, 2019c) and it contributes the following insights:

- Identified 2 types of inefficiencies: complex mode hierarchies and incomplete mode transitions.
- Recommended a strategy to flatten hierarchies by ensuring that one mode is always one tap away from any other mode in the system.
- Recommended a strategy to provide the most versatile mode-switching interaction a virtual keyboard can possibly offer by adopting a dual functionality of both "Persist" and "User-Maintained" techniques.
- Theoretical evaluations reveal 20-60% improvements for currently challenging mobile text entry scenarios like code-switching, password input, mathematical expressions, and programming syntax.

**Maximising Discoverability of SoftCuts**

In Chapter 4, we describe two studies to explore how visual elements of SoftCuts could be optimised. First, we conducted semi-structured interviews to evaluate the usability of three design variants in representing the on-screen keys, showing either the letter only (D1), the command name and a letter (D2), or the command name and an icon (D3). We recruited 12 participants (2 male, 10 female) to use our interactive notetaking and web-browsing app prototype. Each participant was instructed to complete
tasks with all three design variants, one at a time, before specifying their design preference and using a 7-point Likert Scale to rate their experience. Second, we investigated how a modifier key’s saliency and familiarity could affect SoftCuts’ We chose one scenario (note-taking) where the keyboard was always displayed on the screen and another (web-browsing) where the keyboard was hidden under a modifier key by default (i.e. keyboardless). For modifier keys, we used either the Ctrl (familiar to Apple/Windows users) or a completely new and arbitrary design (). We recruited 160 crowdworkers (77 male, 81 female, 1 non-binary, and 1 preferred not to say) to perform a series of tasks, some of which could be achieved only by using SoftCuts. Dependent variables like discovery rates and 7-point Likert rating were used to compare between conditions.

The two studies were published at XXXXXXXXX and they contribute the following insights:

- D3 (command name and icon) is most positively received.
- D2 (command name and letter) is a good alternative for expert users.
- Hotkey usage familiarity boosts the discoverability of SoftCuts.
- Familiarity of modifier key does not affect the discoverability of SoftCuts.
- Saliency of modifier key does not affect the discoverability of SoftCuts.

1.3.2 Empirical Contribution

Evaluating Input Methods for SoftCuts

In Chapter 5, we describe two studies to compare the input methods adopted by commercial systems using soft keyboard shortcuts. Microsoft tablet (PandaSage1221, 2013) adopts two sequential taps (Once) while Samsung tablet (Samsung, n.d.) adopts user-maintained (UM) interaction. The currently discontinued Swype app (Nuance, 2017) adopted sliding gestures (Swipe). For our first study, since we envision SoftCuts as a general command selection mechanism for touch-based devices, we decided to test these three input methods in various configurations, with both mobile phones and tablets, in landscape and portrait orientations, and with one and two hands to perform the interaction. We recruited 12 participants (7 male, 5 female) to select highlighted targets using each input method as fast and accurate as possible. Our second study aims to observe which input method is to be adopted when the user is not constrained and across different activities like sitting, standing and walking. While adoption can be correlated with objective performance, other factors need to be accounted for, such as familiarity with the input method or individual users’ preference. Since the three input methods are compatible with one another, the users have the freedom to use the one they want depending on their preferences for each specific scenario. We are interested in seeing how the first study results would translate when evaluated in a more realistic environment. A new group of 12 participants (7 male, 5 female) was recruited to select the correct key corresponding to each trial’s target command.
Chapter 1. Introduction

The two studies were published at MobileHCI 2020 (Fennedy, Malacria, et al., 2020) and they contribute the following insights:

- Once performs best overall in terms of speed and accuracy
- Swipe is a preferred secondary input method for one-handed phone interaction
- UM is a preferred secondary input method for tablet interaction
- We propose an interaction model where all 3 input methods are compatible with one another

Leveraging Prior Knowledge and Sustaining Retention

In Chapter 6, we describe a study to better understand how the command mappings play a role in maximising user learning. Two types of mappings were chosen: realistic and abstract. Realistic mapping uses potentially known function shortcuts from widely used text-editing application Microsoft Word, like how $\text{A}$ represents the "Select All" function. Abstract mapping uses completely unknown mapping by arbitrarily matching keys with country names, like how $\text{L}$ represents the country "Canada", and are more representative of scenarios where users discover the specific SoftCuts’ mapping on a new app. We recruited 12 participants (4 male, 8 females) to perform command selection as fast and accurately as possible. To evaluate long-term retention, we tested participants’ memory on the position of each command that was tested at multiple phases: at the end of the experiment, after 1, 3, and 7 days.

The study was published at XXXXXXXX and it contributes the following insights:

- There is a minimal performance gap between abstract and realistic mapping: using the latter is only 5% faster, with comparable accuracy.
- Participants leverage prior knowledge with realistic mapping.
- Participants reach an efficient and stable performance level with both mappings.
- Participants rely on spatial memory with both mappings to remember commands’ locations even after a week.

Comparing Performance between Keyboard and Grid Layout

In Chapter 7, we describe a study to compare the performance between a keyboard and a grid layout. To offer a fair comparison, we implemented a full-screen grid layout similar to FastTap (Gutwin, Cockburn, J. Scarr, et al., 2014) and a full-width keyboard layout similar to SoftCuts’ designs used in previous studies on both phones and tablets. 16 participants (8 male, 8 female) were recruited to complete selection tasks using each layout as fast and accurate as possible.

The study was published at XXXXXXXX and it contributes the following insights:
• Selection speed with a grid layout is faster than that with a keyboard layout by 12-16%.

• Selection accuracy is similar between both layouts despite a smaller screen estate occupied by a keyboard layout.

• A keyboard layout supports greater spatial stability than a grid layout.
Chapter 2

Related Work

Before we elaborate on the details from each study, it is essential to have a basic understanding of how previous works reveal SoftCuts’ untapped potential. First, we will look at how the concept of mode-switching has expanded the functionalities of a keyboard beyond its default text-entry capabilities into supporting command selections. Second, we will explore the different strategies proposed to perform command selections on desktop and mobile devices through menus, gestures, and hotkeys. Third, we will show how user learning can be optimised such that skills in using SoftCuts could be acquired efficiently, retained as long as possible, and potentially transferred to other domains of interaction. We conclude this chapter by highlighting key differences between physical hotkeys and SoftCuts and how these differences build our motivation to further investigate and develop features of SoftCuts.

2.1 Mode

This section includes an overview of what mode is about, why we need them, and how they are characterised. Most importantly, we will also show how insights from previous works on mode are relevant to the development of SoftCuts.

2.1.1 Definition

Beyond HCI

In the context of our daily human interaction, *mode* is a prevalent term that has been used to describe a way of life. For instance, we might have witnessed individuals describing their state of minds using terms like ‘party mode’ or ‘serious mode’. For communication, the same message can be represented by different modes: visual, linguistic, spatial, aural, and gestural (Cazden et al., 1996). These examples demonstrate that a mode can be considered a means of achieving something, but it is too general a definition to work with. In this thesis, we will focus our scope of mode on human interaction with physical systems.

Within HCI

In the context of human-machine interface or HCI research, *mode* is not a new topic. Humans have interacted with modes for decades, if not, as Raskin (2000) and MacKenzie (2013) suggested, since forever. One of the earliest attempts to define mode was by Thimbleby (1982), who proposed that a mode is "an interpretation of user input". Poller
Chapter 2. Related Work

14

FIGURE 2.1: An interaction model of a system with one input and three distinct modes, illustrating that mode ‘2’ is responsible for the current blue output. Red and yellow output are associated with mode ‘1’ and ‘3’ respectively.

FIGURE 2.2: Examples of non-computer devices that adopt mode, as listed by Jeff Johnson (1990).

and Garter (1983) echoed this same idea that if the same input to a system can be interpreted in x different ways, there are x different modes available. Subsequent attempts acknowledged this similar understanding of mode but approached it with slight variations: described mode as a state of a system (Brewster, Wright, and Edwards, 1994; Tesler, 2012) or a context of an action (Woods et al., 1994); argued that mode is additionally determined by users’ "locus of attention" (D. Norman, 1981; Raskin, 2000); or framed mode in terms of utility to help users "progressively acquire a skill" (Surale, 2020). Furthermore, researchers have also suggested that we could classify the different types of modes using control theory (Leveson et al., 1997; Degani, 1996) or from the perspective of either the user (action mode) or the system (indicator mode) (Gow, Thimbleby, and Cairns, 2006). Despite this widespread disagreement (J. Johnson and Engelbeck, 1989) about what modes are until today, validating each one of them is not the objective of this thesis. In fact, we recommend looking into Surale (2020)’s work for a more comprehensive study on mode.
Grounding this thesis on a definition provided by Thimbleby (1982) and promoted by Nielsen Norman Group’s Laubheimer (2019), we choose to generalise that moded interaction is observed when the same input to a system generates different output, depending on what the currently active mode is (see Fig. 2.1). Both computer and non-computer devices enable moded interaction. For instance, an automobile system is equipped with a gear stick that can be re-positioned by the driver to control the engine’s output power. Changing gear from ‘drive’ to ‘reverse’ is synonymous with changing a mode as it determines the car’s behaviour when the driver presses the same pedal. Fig. 2.2 shows the non-computer devices that were analysed for moded interaction by Jeff Johnson (1990).

The classic example of moded interaction for computer devices would be a sketching app (see Fig. 2.3a). The same finger swipe on a digital canvas can render different texture, colour, and thickness depending on which tool (or mode) was selected prior. While most of these examples treat mode selection and user input as two separate entities, it is possible to merge both by adopting postures or gestures. For instance, TapSense (Harrison, Schwarz, and Hudson, 2011) allows the same tap action to print different visual output when using different types of finger input like the nail, knuckle, pad, or tip to serve as modes (see Fig. 2.3b). In this case, changing the finger posture before tapping in TapSense is analogous to tool selection (or mode-switching) before stroking in a sketching application.

2.1.2 Significance

The concept of mode was first introduced to address the limited input to support the growing demands for more functions in a system (Thimbleby, 1982; Monk, 1986). This is especially relevant for touch-based interfaces where input is limited to discrete actions like touch down, touch move, and touch release. On the other hand, graphical user interfaces (GUI) could rely on the different buttons available on a mouse and keyboard to activate different modes. This is why interface designers typically rely on
gestures to specify interaction modes for touch-based devices (Isenberg and Hancock, 2012). However, Zhai, P. Kristensson, et al. (2012) highlighted relevant foundational issues regarding gesture designs. One of them is that users would not want to spend time and effort learning new gestures. Mode is beneficial in this case because it facilitates gesture polymorphism (Beaudouin-Lafon and Mackay, 2000) by reusing simpler gestures to make the interface more powerful without making it more complex (Foucault et al., 2014).

However, the root of the problem is not with the presence of modes solely, but particularly the interaction design that facilitates switching from one mode to another, leading to mode errors and confusion (Jeff Johnson, 1990). The lack of contextual awareness in moded systems can have fatal consequences, as seen from the Strasbourg aircraft crash, killing 87 out of 96 people on board (Monnier, 1992). It was reported that the unfortunate crash was because the pilots had mistaken the Vertical Speed mode for the Flight Path Angle mode. Intending to descent at an angle of 3.3°, they entered “33” as input to the system, but unknowingly, they set a high descent rate of 3,300 feet (1 km) per minute. Such negative impacts of mode error surprisingly reach a greater consensus and awareness among the HCI community than the definition of mode itself. Many had proposed strategies to eliminate these errors (D. Norman, 1983) because while an individual mode error may not look harmful, once accumulated, it could result in a bad user experience (Raskin, 2000). With a growing relationship between complex systems and moded interaction (Monk, 1986), we can empathise why Tesler (1981) waged his near-fanatical campaign against the use of mode. However, the benefits of using mode can outweigh its disadvantages if implemented carefully and thoughtfully (Laubheimer, 2019).

2.1.3 Mode-Switching Behaviours

Various interaction techniques have been designed and evaluated in the context of mode (Y. Li et al., 2005; Hinckley et al., 2006; Tu et al., 2012; Surale, Matulic, and Vogel, 2017; Surale, Matulic, and Vogel, 2019; Fenney and H. Lee, 2019a). These techniques can be classified based on the pattern of system behaviours observed while users switch between modes. Dillon, Edey, and Tombaugh (1990) developed the “subtraction method” to estimate the cost of command selection techniques, and Y. Li et al. (2005) suggested that mode-switching is analogous to selecting a command. The mode-switching cost is calculated by measuring the time difference for subjects to complete sequences between two different patterns (Fenney and H. Lee, 2019b): A-A-A-A-A (without mode-switching) and A-B-A-B-A (with mode-switching), where A and B represent tasks or commands from two different modes. We discussed the different techniques in decreasing order of mode-switching cost.

Persist

Persist is by far the most frequently adopted technique in both academic literature and mainstream applications we can find today. Hinckley et al. (2006) defined Persist as a technique that keeps the selected mode active until the user chooses a new mode.
On a sketch app, selecting the pen tool once will persist the pen mode until the user selects a different tool (e.g. brush tool) to persist the new brush mode. This means that once the pen mode has been selected, the user does not need to re-select the pen tool for every stroke operations. *Persist* offloads the mode-maintenance responsibility to the system instead of the users. As a result, a mismatch between the users’ and the system’s mental model often occurs (Fennedy and H. Lee, 2019b), resulting in mode errors. In addition, *Persist* is most costly because the cost is incurred twice. First, to activate the new mode, and the second time is to deactivate the new mode to return to the previous mode (Y. Li et al., 2005). Hence, *Persist* is not ideal for tasks that entail a high frequency of mode-switching.

**Once**

To facilitate tasks with frequent mode-switching, *Persist* technique can be altered to persist the mode selection for a single operation only. Hinckley et al. (2006) named such a technique ‘Once’ while Saffer (2013) named it ‘One-Off’, but we choose to follow the former. *Once* is currently adopted by a popular note-taking app called GoodNotes1. Every time the eraser mode is selected, the completion of a single eraser action (which may include multiple strokes) will automatically return the system to the previous mode. It makes sense to assume that upon deleting a mistake, the user would like to immediately re-enter the correct drawing with whichever mode was selected by the user prior to erasing, and continue where they last paused. *Once* is therefore beneficial and incurs less cost than *Persist* for behaviour pattern A-A-A-B-A-A-C-A that more likely require users to use one primary mode A and persist the secondary mode B and C only once each time.

**User-Maintained (UM)**

For patterns that persist the secondary mode slightly longer than *Once*, *UM* technique can come in handy. It maintains a temporary mode kinesthetically, or only while the user holds a control such as a button or a key Hinckley et al. (2006). Examples can be found on both computer and non-computer devices. For computer devices, holding the Shift button while typing simultaneously will result in the capitalisation of the letters typed. For non-computer devices, when a piano pedal is pressed down by a player’s foot, a sustained or softer sound effect lasts only while the player keeps the pedal pressed down. Releasing either the button or the pedal will return the system to its corresponding default mode: lowercase for keyboard and unfiltered sound for piano. Using our fingers to maintain a mode actively is not a good idea because hands are a scarce resource Jeff Johnson, 1990 when it comes to activities like typing or playing the piano. This is why a foot pedal is chosen for piano instead of using the hand to prevent handicapping the user.

Unlike the previous two techniques *Persist* and *Once*, *UM* transfers the mode maintenance responsibility from the system to the user. Therefore, we follow Sellen, Gordon

---

1https://www.goodnotes.com/
Chapter 2. Related Work

P. Kurtenbach, and W. A. S. Buxton’s approach of generalising such mode behaviour as user-maintained. Alternative names include ‘Quasimode’ (Raskin, 2000; Saund and Lank, 2003) and ‘Spring-loaded’ (Degani, 1996; Hinckley et al., 2006; Saffer, 2013; Pfeuffer et al., 2017). However, the latter may be inaccurate because touch screens do not involve any physical spring in the context of digital devices.

Um technique is made possible by integrating Guiard (1987)’s kinematic chain model for bimanual interaction, where the non-dominant hand can act first to set the context for the dominant hand to carry out precise gestures. This asymmetric bimanual hand coordination helps minimise cognitive and motor costs (Lank, Ruiz, and Cowan, 2006) associated with mode-switching. Although it has been validated by Ruiz, Bunt, and Lank (2008) that the asymptotic cost of adding non-preferred hand modes to an interface is a logarithmic function of the number of modes (Ruiz and Lank, 2007), Raskin (2000) suggested to limit that number to between four and seven. Overall, as supported by W. A. S. Buxton (1995)’s notion of chunking, Um’s continuity of compound tasks effectively reduces mode confusion.

Beyond managing complexity, Um model could be developed further to incorporate a locking mechanism. For example, pianos have an option where users could slide the foot pedal to the left to latch on a small groove. This is useful when fatigue can be caused by a long duration of continued physical exertion to maintain the secondary mode. Such novel features are more difficult to be implemented in physical objects, thus not have been thought of or provided, but are more feasible on software. Found on both physical and soft keyboard, ‘Caps Lock’ offers the system a new state where the secondary mode (i.e. uppercase characters) is given a primary role, but this renders the former primary mode (i.e. lowercase characters) useless. Fennedy, H. Lee, et al. (2018) augmented this by reversing the mode relationship that promotes a flexible swapping between primary and secondary mode.

Pressure

By exerting different amount of force, we can extend Um to facilitate the switching between more than 2 modes. For instance, previous studies used force-sensitive touch screens to provide a more intuitive moded text-entry (Brewster and Hughes, 2009) and text-selection (Goguey, Malacria, and Gutwin, 2018) strategy. However, it is important to note that there have been inconsistent results when using pressure: Corsten et al. (2017) showed over 90% accuracy for 3 pressure levels, while Ramos, Boulos, and Balkrishnan (2004) suggested a maximum of 6 levels. The challenge of using the Pressure technique is in determining intuitive pressure spaces that a diverse spectrum of users can trigger to activate the different modes (Y. Li et al., 2005).

Inferred

One strategy to deal with the mode complexity is by eliminating mode-switching from the users’ end. For instance, a sketching application adopts an interaction protocol that learns the properties and context of pen trajectory before inferring the user’s mode intention and provides the relevant tools automatically. According to Lank, Ruiz,
Chapter 2. Related Work

and Cowan (2006), such a technique is ‘cost-free’ and ideal, because the tasks are indistinguishable with and without mode-switching from the users’ point of view. However, the extent to which a system could make accurate inferences is still questionable. This is possibly why we have not seen Inferred technique being adopted by any commercial system just yet.

2.1.4 Relevance to SoftCuts

The rich history of mode studies deepens our understanding of how moded interaction can be better exploited while designing today’s applications. Mode-switching on a physical keyboard does not enjoy the same benefits as that on a soft keyboard. This is because of the ability to dynamically render appropriate information on the screen may help minimise moded error and communicate command mapping. Particularly in the context of SoftCuts, we need to find out which one of these techniques could effectively facilitate the switching between its primary text-entry mode and its secondary command-selection mode. Within text-entry mode alone, the soft keyboard can be identified to have at least 4 modes (lowercase, uppercase, numbers & punctuation symbols, emojis, additional language, etc) and has adopted Persist with caps lock key, UM with Shift key, and Once with Sticky Key. The interaction complexity of a soft keyboard will only increase as we expand its utility by offering a command-selection mode. Therefore, it is important to study how each of these two modes could influence the overall design of a soft keyboard. Our Chapter 3 will first focus on soft keyboard’s text-entry mode before we dedicate Chapter 4 onward to develop its command-selection mode.

2.2 Command Selection

As defined in Section 1.2.1, command selections are tasks where users select a target item from a bounded set of items (commands). The empirical performance of a selection technique can be measured in terms of its speed and accuracy. For speed, most studies would record when the system first reveals a target to when the user finally makes a correct selection. For accuracy, most studies would treat non-target selections as error or incorrect while counting the percentage of correct selections. The ideal technique should allow users to select targets “as fast and as accurate as possible”. Existing techniques span across different modalities: from mouse-clicking/finger-tapping menu buttons, to drawing stroke gestures using a stylus/finger, and to chording a keyboard shortcut. As a result, the most efficient way to invoke a command depends on the platform used. For instance, to trigger Undo, an Android phone user would rely on tapping menu items while an iOS phone user would rely on a shaking gesture while a desktop computer user would rely on the chording of $\text{Ctrl}+\text{Z}$. In this section, we will focus on desktop-based and touch-based techniques while excluding that of mid-air or those that have been covered in the previous Section 2.1. We recommend looking at the extensive review of visual menu techniques by Bailly, Lecolinet, and Nigay (2016).
Chapter 2. Related Work

2.2.1 With menus

Most GUIs on desktop personal computers (PC) support interactivity through widgets like windows, icons, menus, and pointers (WIMP). Double-clicking an application icon using a mouse pointer will launch the application and contain it within a window. The available commands to an application are made available through different menu approaches (see Fig. 2.4). However, applications can quickly run out of screen space if they were to persistently and linearly feature the names of all their commands. One solution is to semantically group commands through drop-down menu hierarchies and feature only the names of parent items in a menu bar. Contextual menus can also be used to filter commands based on the object selected prior. Alternatively, toolbars often substitute lengthy command names with familiar and succinct visual encoding (i.e., icons). Ribbons replace both traditional menu bars and toolbars by grouping icons semantically in tabs and are commonly used in Microsoft Word applications (Microsoft, 2020).

These menu approaches are similar in that users do not have to recall from memory but rely on visual recognition instead (K. L. Norman, 1991). While such a strategy offers advantages to novices who may not be familiar with how items are organised, it unfortunately disadvantages experts. This is because despite having prior knowledge of target locations, experts are forced to take additional steps navigating through the hierarchy (Cockburn and Gutwin, 2009). For instance, CommandMaps (J. Scarr, Cockburn, Gutwin, and Bunt, 2012) flatten menu hierarchies by leveraging spatial memory to boost expert performance. As shown in Fig. 2.5a, application commands are displayed all at once in a spatially stable fashion to maximise long-term retention. J. Scarr, Gutwin, et al. (2015) extended this CommandMaps idea by highlighting command subsets to form StencilMaps and EphemeralMaps. StencilMaps use a dark semi-transparent ‘stencil’ overlay to de-emphasise all but the subset items (see Fig. 2.5b) while EphemeralMaps use a short delay, with subset items shown immediately, and
Chapter 2. Related Work

FIGURE 2.5: Spatially stable command selection interfaces: (a) CommandMaps from J. Scarr, Cockburn, Gutwin, and Bunt (2012), (b) StencilMaps and (c) EphemeralMaps from J. Scarr, Gutwin, et al. (2015).

FIGURE 2.6: Food-inspired user interface (UI) elements for menu navigation design on mobile devices.

other items gradually faded in (see Fig. 2.5c). However, it is evident that strategies like CommandMaps, StencilMaps, and EphemeralMaps consume a lot of screen space which can be a limited resource in smaller devices or certain applications.

On the other hand, today’s mobile devices are equipped with touchscreen interfaces that rely on gestures or direct manipulation with icons and menus. Lacking in windows and pointers, this interface is referred to as "post-WIMP" by Van Dam (1997). Screen real estate becomes increasingly limited as we shift our attention from desktop to tablets and smartphones. Designers had to be creative with this limitation and hence, prioritise the most frequently used commands such that they are accessible without much user effort. Using MS Word application as an example, Fig. 2.7 and 2.8 demonstrate how commands are grouped in subsets of varying layout and can then be revealed by tapping respective menu icons. This means that novices will have to visually search for the desired command within and between layers of the menu hierarchy, which negatively affects usability (Pernice and Budiu, 2016). If visual search leads to no avail, one could tap on the light bulb icon (Fig. 2.8c) to reveal a command search bar. However, most mobile apps today, unfortunately, do not support users in such feature discovery (Alvina, Bunt, et al., 2020).

Similar to desktop platforms, previous studies on expert touch-based command selections have explored various visual layouts to exploit spatial memory (see Fig. 2.9). For instance, FastTap (Gutwin, Cockburn, J. Scarr, et al., 2014) overlays a full-screen
Chapter 2. Related Work

Figure 2.7: Hierarchical navigation of commands in Microsoft Word tablet application. Tapping on the meatballs menu (a) will reveal more commands like (b), which act as sub-menus to more commands in (c).

Figure 2.8: Hierarchical navigation of commands in Microsoft Word smartphone application. Selecting a text element will reveal a keyboard and a scrolling toolbar (a), which features a subset of commands. The full set (c) can be accessed by either tapping on the meatballs menu in (a) or the text-editing icon (b). However, tapping the meatballs menu (d) will reveal a different set of commands, like (e) and (f), related to the whole document instead of a specific text element.
grid layout on smartphones to support chording between thumb and forefinger. This technique not only enables rapid command selection (i.e., fewer actions and lesser movement) but also supports a large number of commands, so much so that a modified version of FastTap was proposed for much smaller screens of smartwatches (Lafreniere, Gutwin, Cockburn, and Grossman, 2016). As shown by HandMark Menus (Uddin, Gutwin, and Lafreniere, 2016), the same grid layout can also be framed around readily available and familiar landmarks like users’ hand and fingers. While the efficacy of this technique is currently only evident for tabletop interactions, it is a promising strategy to be incorporated on tablets as well. Alternatively, bezels, which are borders between a screen and a device’s frame, can also be used to organise command items. Schramm, Gutwin, and Cockburn (2016) demonstrated that swiping or double-tapping hidden toolbars offer under-exploited performance benefits that combine spatial memory with rehearsal to enable smooth transitions to expertise.

Using keyboard layouts

Besides the hierarchical, grid-based, or bezel-based menu layouts, the under-utilised keyboard layout can be an alternative strategy to organise command items and facilitate selections. A standard keyboard has at least 78 keys, the majority of which are primarily designed to serve text entry. Its physical properties support the tactile acquisition of keys such that users can touch-type at high speed and accuracy and home their
fingers easily. The same benefits can be extended to command selections, as shown by previous works (see Fig. 2.10). For instance, Block, Gellersen, and Villar (2010) embedded touch-sensing capabilities and projected dynamic display in each key to transform a standard keyboard into an interactive space. A similar idea was commercially realised by Optimus Popularis keyboard (Studio, 2011), allowing users to customise their preferred mapping between keys and functions. The keyboard layout can also be rendered virtually at the bottom half of the computer screen (Berman and Hourcade, 2014) to leverage D. A. Norman (2002a)’s principle of natural mapping. Aceituno and Roussel (2014) presented how Hotkey Palette can deliver quick retrieval of documents and effective management of windows using the stable layout of a virtual keyboard.

As discussed above, menu techniques may be relatively easy to use for novices on desktop devices. However, it becomes more challenging for mobile users who have yet to discover hidden commands or navigate hierarchical layers due to limited screen space. A more efficient strategy is to use stable layouts like that of grid and keyboard to leverage users’ spatial memory. The challenge with this strategy is that the mapping between spatial position and command items is not standardised across applications. Therefore, keyboard shortcuts (or better known as hotkeys or accelerators) like cut, copy, and paste commands, represent a more promising and consistent approach to be discussed in Section 2.2.3. More details about spatial memory will also be discussed in the context of skill acquisition in Section 2.3.

2.2.2 With gestures

As defined in Section 1.2.1, a gesture can be referred to as a sequence of human actions executed automatically when set in motion. Bailly, Lecolinet, and Nigay (2016) referred to them as “stroke shortcuts”, but this might limit gestures afforded by a stylus/pen. Instead, we also consider gestures performed by our bare hands or chorded by our fingers, which are relevant for multitouch interaction from smartphones to tabletops. Examples of gestures include pressing, tapping, hovering, holding, swiping, or strokes, which are standard gestures we can find on today’s touch-based devices (Apple, n.d.; Pong and Malacria, 2019).

are facilitated by drawing straight lines in either menu mode or mark mode (see Fig. 2.11a). In menu mode, users who are unfamiliar with the gesture mechanism could press the pen against the display and wait for approximately 1/3 of a second (Gordon Paul Kurtenbach, 1993). A radial menu then appears directly below the tip of the pen for users to draw a line towards the sector of the desired item. When the user lifts the pen up, the selected item is then highlighted as feedback. In mark mode, users could immediately draw the strokes without revealing the menu if they are expert enough to recall the mapping between a stroke sequence and a command. Several variations on the general MM approach have been proposed, including FlowMenu (Guimbretière and Winograd, 2000), Flower Menus (Bailly, Lecolinet, and Nigay, 2008), and Multi-Stroke Marking Menus (Zhao and Balakrishnan, 2004; Zhao, Agrawala, and Hinckley, 2006). Studies have shown radial layouts like MM are faster and more accurate than conventional linear menus (Fitts, 1954; Callahan et al., 1988), and radial layouts can exploit spatial memory by associating commands with cardinal orientations (Bailly, Lecolinet, and Nigay, 2016). MM have also been extended to touch (Zheng, Bi, et al., 2018) and mid-air (Ren and O'Neill, 2012; Gebhardt et al., 2013; W. Li et al., 2021) interactions.

However, MM are limited to gestures composed of straight lines, and they do not support arbitrary gesture shapes such as curved paths, letters, or symbols (Goldberg and Richardson, 1993; Wobbrock, Wilson, and Y. Li, 2007). FlowMenu and Flower-Menus are variations of MM that adopt curved paths. Crib-sheet (G. Kurtenbach, T. P. Moran, and W. Buxton, 1994) supports the execution of arbitrary gestures by displaying them with their corresponding commands in a static list or a grid (see Fig. 2.11b). If a user finds a particular gesture complicated to replicate, pressing the gesture icon will reveal a textual description and play an animated demonstration of the gesture execution. A more dynamic implementation would be Bau and Mackay (2008)'s OctoPocus (see Fig. 2.11c). Also activated with a dwell trigger, it renders all possible gesture paths (or templates) emanating from the current input location. Each path is coloured and labelled, and they are dynamic because paths are filtered as the gesture is partially completed. Less likely ones fade away and eventually disappear while the most likely ones remain. Roy et al. (2013) also explored combining both unistroke letters with MM to form Augmented Letters (see Fig. 2.11d). It leverages mnemonic mapping by first requiring users to draw the first letter of a command name before appending a directional tail (mark) to handle conflicts between commands that share the same initial letter.

Gestures are not limited to only lines drawn on a surface, especially when recent studies have demonstrated how users' hands can play a more active role in forming more advanced gestures (see Fig. 2.12). For instance, ShadowGuides (Freeman et al., 2009) first visualise the registration pose guide so that users are aware of all available hand postures in the system. Then, it uses shadow annotations to animate how the hand can be moved from the current hand pose. On the other hand, MelodicTap (Heo, Jung, and G. Lee, 2016) and Arpège (Ghomi et al., 2013) leverage our fast fingering skills to make a sequential tap gesture, similar to the way we play the piano/guitar. The main difference is in the graphical interface: MelodicTap renders rectangular keys (like in a piano) while Arpège renders coloured circles (which mimic fingerprints) to represent items in a particular menu hierarchy layer. A very different approach can be found in
Chapter 2. Related Work

Figure 2.12: Techniques using hand gestures: (a) ShadowGuides (from Freeman et al. (2009)), (b) MelodicTap (from Heo, Jung, and G. Lee (2016)), (c) Arpège (from Ghomi et al. (2013)), and (d) Gesture Play (from Bragdon, Uguray, et al. (2010)).

Bragdon, Uguray, et al. (2010)’s Gesture Play. It uses physical metaphors like buttons, animated springs, and wheel props to disclose static and dynamic hand gestures. These examples represent just how expressive a single hand can be for executing gestures. The other side of the coin is that there is no standardised gesture dedicated for a given command across applications or platforms.

Using keyboard layouts

Similar to menus (Section 2.2.1), gestures also leverage keyboard layouts to contextualise the strokes drawn by the user. This strategy is leveraged by a speed-writing recognition system called SHARK, which augments both stylus keyboarding and shorthand gesturing (Zhai and P.-O. Kristensson, 2003; P.-O. Kristensson and Zhai, 2004). For instance, Command Strokes (P. O. Kristensson and Zhai, 2007) renders a virtual keyboard and requires users to use a stylus and draw a stroke from a modifier key through a few or all letters in the name of a command (i.e. C-O-P-Y for a “Copy” command, see Fig. 2.13a). We also found a similar strategy adopted by HotStrokes (Cui et al., 2019) and GestKeyboard (Zhang and Y. Li, 2014). HotStrokes uses a laptop trackpad as a canvas, while GestKeyboard uses any physical keyboard and its gesture set form symbolic shapes like circle, rectangle, and pigtail (see Fig. 2.13d). Since gesture-typing had been incorporated by smartphone devices, CommandBoard (Alvina, Griggio, et al., 2017) took this opportunity to combine a keyboard layout with an OctoPocus-style (Bau and Mackay, 2008) or radial menu to invoke commands. As shown in Fig. 2.13c, a user could first swipe through the words like “happy” or “color”. Subsequently, suppose the user slides his/her finger into the space above the keyboard, the respective menu

\[\text{https://www.androidauthority.com/android-4-2-jelly-bean-official-here-what-you-need-know/}\]
Chapter 2. Related Work

Figure 2.13: Techniques using keyboard layouts to frame stroke gestures: (a) Command Strokes (from P. O. Kristensson and Zhai (2007)), (b) HotStrokes (from Cui et al. (2019)), (c) CommandBoard (from Alvina, Griggio, et al. (2017)), and (d) GestKeyboard (from Zhang and Y. Li (2014)).
will be rendered to format text or to change the color of the currently active tool, depending on what word was gestured prior.

Gestural techniques are known for their fast invocations (P. O. Kristensson and Zhai, 2007). However, unlike menus, gestures are not self-revealing by default (Baudel and Beaudouin-Lafon, 1993; Pong and Malacria, 2019), so users need to explicitly discover, learn, and memorise them. Avery and Lank (2016) validated this perception of the slow adoption of gestures. They determine that users have a relatively high awareness of expert-level gestures and willingness to perform them but often find them difficult to discover and challenging to learn. This goes against many usability guidelines (Nielsen and Molich, 1990) and even prompted D. A. Norman and Nielsen (2010) to characterise gestures as “a step backward”. In addition, the mapping between a gestural input and a command output is invisible, inconsistent, and unnatural (D. A. Norman, 2010), hence prompting users to continuously rehearse and relearn new mappings as they switch between applications or devices. Therefore, we need a more versatile command selection techniques that not only raise the performance ceiling of experts but also provide sufficient support for novices to gain expertise over time.

### 2.2.3 With hotkeys

Hotkeys are keyboard keys which when pressed sequentially or simultaneously would invoke the corresponding command mapped to that particular key combinations and order. They facilitate rapid access to specific commands by pressing a modifier key(s) in conjunction with character key(s). Examples of a modifier key include \[\text{ fled }, \text{ ctrl }, \text{ alt }, \text{ option }, \text{ or } \text{ shift}.\] They are designed to disambiguate between the keyboard’s text-entry mode and command-selection mode when the character key is subsequently pressed. While modifier-less hotkeys exist, typically in graphical applications such as Adobe Photoshop or GIMP, they remain much less frequent than hotkeys relying on modifiers. Therefore, we focus on this common type of hotkey in this thesis. For instance, commands like cut, copy, and paste are associated with first pressing the modifier key / \[\text{ ctrl }\] and then \[X, C, \text{ and } V\] character key respectively, and have been established universally.

From a visual perspective, hotkeys can be considered as menu items arranged in a keyboard layout, as highlighted in Section 2.2.1. The main difference is that hotkeys share a common mapping that has been standardised across many (not all yet) applications and desktop OS. While users could personalise their favourite hotkey bindings, such a feature is usually presented with caution so that it does not result in confusion/conflict with those that have been reserved or widely accepted by users. Hotkeys are also similar to gestures in that they both are faster command selection mechanisms than menus (Odell et al., 2004). This could be the reason why hotkeys are also known as “keyboard shortcuts”. Theoretical models like Keystroke-Level Model (KLM) (Card, 3

---

3[https://developer.apple.com/design/human-interface-guidelines/macos/user-interaction/keyboard/]
Chapter 2. Related Work

Figure 2.14: Strategies to boost hotkey usage: (a) Skillometers (from Malacria, J. Scarr, et al. (2013)), (b) ExposeHK (from Malacria, Bailly, et al. (2013)), (c) IconHK (from Giannisakis et al. (2017)), and (d) KeyMap (from Lewis, d’Eon, et al. (2020)).

Thomas P. Moran, and Newell, 1980) also demonstrated that hotkeys improve user performance as compared to menus. However, Appert and Zhai (2009) conducted a study between two shortcuts to menu selection: hotkeys and stroke arbitrary gesture shapes. They showed that when there is no mnemonic link between the shortcut and the command, users could recall more shortcuts with gestures. The lack of consistent standards (D. A. Norman, 2010) in gestures has resulted in its low adoption (Avery and Lank, 2016) as compared to hotkeys.

Despite their benefits, hotkeys are not without their limitations. First, users need to know the key combination prior to triggering hotkeys, and hence novice users are at a direct disadvantage. Second, according to Lane et al. (2005), most experienced users rarely used efficient keyboard shortcuts, favouring icon toolbars instead. This tendency of users to stick with procedures they already know regardless of their efficacy has serious implications for the productivity of millions of office workers (Cockburn, Gutwin,
et al., 2014). Previous studies have proposed various strategies to increase hotkey accessibility and usage. One strategy can be seen in Skillometers (Malacria, J. Scarr, et al., 2013). It builds user motivation by raising user awareness of the benefits of switching to a faster mode of interaction through visualising their past and potential future performance (see Fig. 2.14a). Alternatively, ExposeHK (Malacria, Bailly, et al., 2013) overlays hotkeys onto existing widgets when a modifier key is pressed (see Fig. 2.14b). It aims to facilitate hotkey browsing while simultaneously supporting physical rehearsal. It has been incorporated into many applications, including Slack⁴, which is a team communication and collaboration app optimised for workplace productivity. We also found IconHK’s (Giannisakis et al., 2017) approach to communicating keyboard shortcuts novel. It blends visual cues into toolbar buttons without denaturing the pictorial representation of their command. On the other hand, KeyMap (Lewis, d’Eon, et al., 2020) moves away from using toolbar and instead render a virtual keyboard layout with command labels displayed directly on its keys (see Fig. 2.14d) to leverage Norman’s principle of natural mapping (D. A. Norman, 2002a). More strategies will be discussed in Section 2.3, focusing on the impact of keyboard shortcuts on users’ skill development.

There have also been increasing attempts to expand the potential of keyboard shortcuts by combining gestural techniques to increase user input vocabulary. Finger-Aware shortcuts (Zheng and Vogel, 2016) harnessed the benefits of mode (see Section 2.1) by demonstrating how the same key can be mapped to different command output when

---

⁴https://slack.com/
the keypress is combined with various finger/hand posture (see Fig. 2.15a). Zheng, Lewis, et al. (2018) supported this idea by providing dynamic visual guidance (see Fig. 2.15b) in FingerArc and FingerChord to reduce the gulf between graphical input and shortcut activation. However, most other techniques were not equipped with such a very much needed guidance as users would not be able to recall the larger vocabulary mapping. For instance, Bailly, Pietrzak, et al. (2013) proposed raising individual keys to support new user input like squeezing and directional pushing in addition to the default pressing and releasing (see Fig. 2.15c). Alternative strategies to define a new class of hotkeys include altering the order of keys pressed (Au, Yeung, and Cheung, 2016) and combining with arm and wrist rotations (Buschek, Roppelt, and Alt, 2018). While these techniques push the envelope of hotkeys, it remains challenging for integration into practical applications and user adoption.

2.2.4 Relevance to SoftCuts

In this Section 2.2, we explored how command selections on desktop and mobile devices can be facilitated through menus, gestures, and hotkeys. Of particular interest, both menu-based and gesture-based interaction exploited keyboard layouts. This is partly due to the ubiquity and versatility of keyboards. Desktop computers have physical keyboards, while mobile devices have transient soft (virtual or on-screen) keyboards. One thing is clear: keyboard presence will stay for quite some time, be it hard or soft. Therefore, this is an excellent opportunity for SoftCuts to leverage users’ familiarity through years of keyboard’s established presence. Although attempts to instantiate SoftCuts can be found on some commercial tablet devices like Samsung Galaxy (Samsung, n.d.) and Microsoft Surface (PandaSage1221, 2013) (see Fig. 1.4), its full potential has yet to be realised. For instance, there is no consistency in terms of visual representations or input methods for SoftCuts, which we will investigate further in Section 4 and 5. It is also unclear how SoftCuts’ designs should be optimised for smartphones, which has a larger user base and more limited screen real estate than tablets.

2.3 Skill Development

According to Merriam-Webster dictionary\(^5\), the term *skill* can be defined as the ability to use one’s knowledge effectively and readily in execution or performance. Skill covers a large range of phenomena from mathematical abilities to driving or running (Adams, 1987). In this section, we are going to focus on human factors of skill development particularly relevant to UI designs, from desktop to mobile devices:

- How can we help users to accelerate their transition from being a novice to an expert?
- How can we help users to retain their expert performance in the long run?
- How can we help users to apply their expert skills from one domain to another domain?

\(^5\)https://www.merriam-webster.com/dictionary/skill
These questions will be addressed in the following subsections, where we dig deeper into how learnability\(^6\) can be designed to maximise usability (Nielsen, 1994; Shneiderman, 1997).

### 2.3.1 Skill Acquisition

Training is essential to improving human performance. Early works (Richard A Schmidt, 1988; Krakauer and Mazzoni, 2011) suggested that the performance of our motor skills can be improved through guidance and practice over time. This improvement can be characterised qualitatively by adapting the *power law of practice*, which was first observed by Snoddy (1926) and then generalised by Fitts (1964) and Newell and Rosenbloom (1981). It considers the learning curve effect on performance by stating that the logarithm of the reaction time for a particular task decreases linearly with the logarithm of the number of practice trials taken. The black and blue power curves in Fig. 2.16 represent the *intramodal* expertise development of two different modalities. Based on the framework proposed by J. Scarr, Cockburn, Gutwin, and Quinn (2011), each curve can be further divided into three segments: initial performance, extended learnability, and ultimate performance. The framework is analogous to Fitts and Posner (1967)'s three-stage model, which identifies how users’ cognitive models are first formed, followed by how the associations between concepts are then established over time and concludes in the development of autonomous skills. However, ‘practice makes perfect’ is a critical training fallacy identified by W. Schneider (1985). High-performance training (e.g. in air traffic control) was quoted to demonstrate a slow acquisition rate despite dedicating a considerable portion of the training time to practising the task. It is vital to first understand the nature of the task before designing the appropriate training structure and approach to guarantee maximum performance growth.

\(^6\)https://www.nngroup.com/articles/usability-101-introduction-to-usability/
As shown in Section 2.2, there could be more than one modality (or input method) to trigger the same command output. Menus, gestures, and hotkeys offer users different performance floor and ceiling, and switching between them to pursue a higher ceiling brings about intermodal expertise development. The first stage of supporting intermodal transition is to raise awareness of the new modality. For instance, traditional menus often feature hotkey labels, as shown in Fig. 1.1b and 2.4. Another example is Blur (J. Scarr, Cockburn, Gutwin, and Quinn, 2011) which uses the calm notification and hot commands to support a transition from WIMP interaction to a more efficient command-based interface. The second stage is to communicate a positive perception of the new modality. Despite clear evidence that hotkeys perform faster selections than using mouse and toolbar (Odell et al., 2004; Lane et al., 2005), Tak, Westendorp, and Rooij (2013) found that some of their participants did not use hotkeys because they believed the opposite is true. A lightweight display like Skillometers (Malacria, J. Scarr, et al., 2013) could help address this problem by revealing the meta-level understanding of user interaction. As shown in Fig. 2.14a, Skillometers could indicate how much time spent by clicking a mouse and compare it with an estimated keystroke time for the associated hotkey. The amount of time saved is one benefit to encourage users to use more hotkeys. However, knowing the existence and benefits of the new modality is not sufficient to guarantee that users will eventually make the switch (Cockburn, Gutwin, et al., 2014). According to Bhavnani and John (1997), even experienced computer users still use inefficient strategies. Grossman, Dragicevic, and Balakrishnan (2007) suggest accelerating the transition by manipulating feedback and cost associated with a given input modality. A similar approach is found in HotKeyCoach (Krisler and Alterman, 2008), which shows a dialog box whenever an item is selected with the mouse, requiring that either an extra click be used to proceed or the hotkey be invoked (see Fig. 2.17b). Although such methods prove successful in increasing hotkey use, forced use of hotkeys or audio feedback may not be accepted outside lab settings.
performance dip (see Fig. 2.16) when switching to a new modality, even if it offers a higher ultimate performance ceiling. While the dip may only be temporary, most users are reluctant to make the switch and tend to reuse existing or known method. This tendency is best explained by Carroll and Rosson (1987)’s paradox of active user, which suggests interface performance to reach an asymptotic level of mediocrity. The dip size is also influenced by the magnitude of difference between the input modalities (Cockburn, Gutwin, et al., 2014). A fundamental principle to minimise performance dip is Gordon Paul Kurtenbach (1993)’s idea of rehearsal – that the physical actions taken by a novice to select a command should be the same as those taken by an expert. As the novice learns the commands, they can start to make selections using the expert method, which is faster because there is no need to wait for feedback (see mark mode in Fig. 2.11a). Let us examine at how expert methods like gestures and hotkeys minimise performance dip through guidance and rehearsal.

In principle, a gesture or a hotkey does not require any visual interface: the stroke or chording action is performed, and the associated command is triggered. Of course, this assumes the user already knows how to perform the action and its command mapping. This is why gestures and hotkeys are commonly associated with expert usage, while novices initially rely on a guide. A guide could be an offline user manual or tutorial video. However, we focus on interaction techniques that have guidance already integrated within it to support all users regardless of expertise.

A guide help users to learn complex interfaces in two ways: feedforward and feedback. First, a guide provides feedforward (Djajadiningrat, Overbeeke, and Wensveen, 2002) by revealing all possible actions and their command mappings to users. Feedforward bridges the difference between user’s intentions and the allowable actions (Vermeulen et al., 2013), also known as D. A. Norman (2002b)’s Gulf of Execution. Crib-Sheet (G. Kurtenbach, T. P. Moran, and W. Buxton, 1994) (see Fig. 2.11b) is one example of a static feedforward, which means that the guide is independent of any partially executed input. It can be enhanced to show each gesture’s text descriptions and animated renderings, like in GestureBar (Bragdon, Zeleznik, et al., 2009) (see Fig. 2.17a). For dynamic feedforward, we can see from OctoPocus (Bau and Mackay, 2008) in Fig. 2.11c, that the style and scope of the display continuously update based on user input for a partially completed gesture. Second, a guide may include feedback. Unlike feedforward, which is offered before the action occurs, feedback is communicated during or after the action. It bridges D. A. Norman (2002b)’s Gulf of Evaluation: the effort users must exert to interpret the system’s state and determine how well the expectations and intentions have been met (Vermeulen et al., 2013). Feedback can also be static or dynamic. Static feedback is displayed after gesture execution, indicating what gesture was recognised, possibly with a level of confidence. On the other hand, dynamic feedback is provided during execution to communicate information like the current level of recognition confidence based on a partial gesture. However, Delamare, Janssoone, et al. (2016) revealed how dynamic feedback has higher recognition accuracy than static feedback when evaluated with their 3D variant of OctoPocus. We recommend looking at Delamare, Coutrix, and Nigay (2015)’s work for a more comprehensive literature review of feedforward and feedback mechanism.
Spatial memory

One aspect of human memory that directly influences skill development with UIs is spatial memory. Studies have shown that spatial memory not only helps people remember many items (Robertson et al., 1998), but it also persisted over several months (Czerwinski et al., 1999). In Section 2.2, we included how desktop and mobile devices could leverage spatial memory. CommandMaps (J. Scarr, Cockburn, Gutwin, and Bunt, 2012) with its menu ribbon layout, MM (Gordon Kurtenbach and B. Buxton, 1994) with its radial layout, FastTap (Gutwin, Cockburn, J. Scarr, et al., 2014) with its grid layout, and also hotkeys with its keyboard layout. This is because J. Scarr, Cockburn, Gutwin, and Malacria (2013) provided evidence that spatial stability is robust to view transformations like scaling and perspective change. We recommend looking into J. L. Scarr (2014)’s work for a more comprehensive review on spatial memory.

Incidental Learning

Psychology literature has demonstrated how our memory can be improved with incidental learning (Shelton and Newhouse, 1981). McGeoch (1942) defined incidental learning as a type of learning that occurs without a specific motive or instruction to learn the material in question. Asselen, Fritschy, and Postma (2006) reported studies on how intentional and incidental learning influenced the depth of spatial knowledge acquisition differently during navigation. In the case of HCI, it suggests that users could learn interface components as a side effect of their display while using other components. For instance, training a user to select a command from a given layout may expose him/her to neighbouring commands sharing the same region. Beyond a menu layout, gestures can also facilitate incidental learning when combined with a feedforward guide. For instance, in our recent work [Kat: still under revision for TVCG], we showed how OctoPocus (Bau and Mackay, 2008) could be adapted for the immersive virtual reality (VR) environment by exploiting the 3D spatial memory of users. This prior exposure benefits future invocations as the user is now aware of its availability and can estimate the area of space to narrow down for visual search. This is why incidental learning not only increases input vocabulary but also speeds up selections.

2.3.2 Skill Retention

Drawing from examples of motor and verbal tasks, Richard A. Schmidt and Bjork (1992) showed that performance during practice is not indicative of learning but only evident through retention and transfer tests. This is further confirmed by Cockburn, P. O. Kristensson, et al. (2007) who revealed that although training condition requires more mental effort and slow down user performance, it is the subsequent testing condition that demonstrates a different result. The guidance hypothesis by Richard A. Schmidt (1991) suggested that augmented feedback which improves early performance through guidance, may impair retention of the performed skills once the guidance is
Figure 2.18: TouchTools designed virtual tools (top) to replicate the real-world grasps (bottom). Image is obtained from Harrison, Xiao, et al. (2014).

removed. Therefore, it is equally important to understand how interfaces could be designed to maintain user expertise in the long run, especially when guidance is no longer available or when memory and motor skills decay over time.

For instance, F. Anderson and Bischof (2013) proposed an adaptive guide that is not detrimental to user learning. In the first few trials, a static tracing guide was provided so that participants could execute gestures with high accuracy and usability. Midway through the trials, the guide only disappears when user stroke reaches half the gesture template length. In the last few trials, the guide never appears at all. Adaptive guide progressively forces the user to draw the gesture themselves without relying on any guidance. However, such implementation in the real world is potentially more difficult because the number of trials is unknown, unlike that in a controlled lab study. In addition, for post-training persistence, Lafreniere, Gutwin, and Cockburn (2017) concluded that participants exhibited a somewhat surprising “all-or-nothing” effect, using the expert technique nearly exclusively or not at all. This suggests that switching to an expert technique involves a global change by the user rather than an incremental change (Fu and Gray, 2004; F. Anderson and Bischof, 2013).

2.3.3 Skill Transfer

Skills learned in one context can be transferred to a new and different context. In the Reality-Based Interactions (RBI) framework, Jacob et al. (2008) exposed the benefit of using pre-existing skills to reduce the cognitive load and the learning cost of the interaction. This idea also appears in one of Wigdor and Wixon (2011)’s design guidelines for natural user interfaces, which is to “leverage innate talents and previously learned skills” One of the examples that harness the benefits of skill transfer is TouchTools (Harrison, Xiao, et al., 2014). It leverages user familiarity and skill with physical tools (e.g. eraser, marker, or camera in Fig. 2.18) to augment touch interaction on a tablet.

Bérard and Rochet-Capellan (2015) assessed the practical value of sensory-motor skill transfer between the interaction with physical and digital objects in a serial target acquisition task. They found out that the learning of a novel task with direct-touch interaction transferred very well to physical interaction (i.e. with a physical disc). From the black arrow in Fig. 2.19a, we can see that transfer is assessed by comparing the
Chapter 2. Related Work

performance in this last block with the baseline performance. If the effects were negative, it will be called interference. Henderson, Mizobuchi, et al. (2019) also observed similar cross-modal benefits when they leveraged touch gesture input to teach users to perform mid-air gestures. Most recently, Raissi et al. (2020) investigated the impact of similarity and number of alternations between two abstract keyboard layouts on retroactive transfer. A retroactive transfer is the influence of the new skill on the acquisition of a previously learned skill (Bunch, 1946; Postman, 1971). In Fig. 2.19b, the temporal performance drop (TPD) represents the temporal difference between the end of one method and the beginning of another method. We can see that the amplitude of the drop decreases with the number of alternations. These previous studies reveal the benefits of leveraging prior knowledge by incorporating design layouts and motor skills as familiar as possible for the users.

2.3.4 Relevance to SoftCuts

From this Section 2.3, we learn a lot about how SoftCuts could maximise its usability by minimising the effort needed to both learn as a novice and maintain skill as an expert. First, SoftCuts need to accommodate novices who might not be familiar with hotkeys on desktop systems. That is why we will investigate different visual design strategies in Chapter 4 and share not only how to make SoftCuts easy to discover as a feature, but also facilitate feedback for all the commands available. Second, unlike rehearsal-based interfaces that rely on time delay to increase expert usage, we will not be implementing any delay for SoftCuts. This is because recent studies (Henderson,
Malacria, et al., 2020; Lewis and Vogel, 2020) have shown that delays can increase error rate, and its necessity has been questioned in the context of MM. Third, SoftCuts’ keyboard layout provides opportunities for users to exploit their spatial memory in both intentional and incidental learning. In Chapter 6, we will study how our memory retention could be affected when we vary the mapping between a spatial area and command output. It is also rather unclear how would a keyboard layout fare with a more standardised grid layout like that in FastTap (Gutwin, Cockburn, J. Scarr, et al., 2014). We will study the performance difference between these two spatially stable layouts in Chapter 7.

2.4 Why SoftCuts?

In this chapter, we have identified key related works that span across mode-switching techniques, command selection strategies, and last but not least, user skill development. With modes, a keyboard designed primarily to serve text input can now support command selections via hotkeys. On a physical keyboard, hotkeys may be associated with expert users because their mappings being hidden in menus require users to rely on their memory. In the case of SoftCuts, the soft keyboard’s ability to render dynamically helps communicate the mapping information needed by novices or experts alike. Beyond ensuring an efficient command selection accessible to users regardless of expertise, the keyboard layout leveraged by SoftCuts could answer the lack of standardised command mapping offered by menus and gestures. However, the downside of a soft keyboard is that it is only transient due to limited screen real estate and there is no (passive) haptic feedback, unlike a physical keyboard, where it is common to observe users reliably chord hotkeys eyes-free. Therefore, in the following chapters, we elaborate the studies to confirm the feasibility and benefits of SoftCuts in transferring hotkey skills and learning from a desktop keyboard to a soft keyboard. All in all, these related works have demonstrated the potential of hotkeys to benefit mobile devices by capitalising on a soft keyboard. Hence, the shift in the hotkey environment that SoftCuts is proposing not only makes sense but also a timely one.
Chapter 3

Optimising Moded Interaction Design for Soft Keyboard

Fundamentally, SoftCuts is a command selection mechanism that leverages the familiar keyboard layout whose primary function is for text entry. In this Chapter, we choose to analyse the mode-switching behaviour observed while typing on a soft keyboard before delving deeper into its command-selection mode in the subsequent chapters.

3.1 Motivation

Despite how the QWERTY layout (D'Cruz, 2014) has commercially stood the test of time against alternative proposed keyboard designs (Bi, Smith, and Zhai, 2010; Dunlop et al., 2012; Joshi et al., 2011), no prior study has paid much attention to the arrangement of its supporting elements (e.g. numbers, punctuation, symbols, modifier keys or secondary languages). The problem is because the same layout used on a physical keyboard can no longer be accommodated in the limited screen real estate of our smartphones and tablets. For every additional space dedicated to the keyboard, we may be able to present more keys. However, the tradeoff is that less space is then available to present the application-relevant data and interaction. Therefore, sacrifices had to be made by prioritising a specific group of keys (i.e. lowercase/uppercase characters) in the default display (see Fig. 1.2) while the remaining keys can then be accessed by switching modes.

The usage frequency of non-default keys may be observed to be relatively lower than that of default keys. However, this does not mean that accessibility could take a back seat for non-default keys. In fact, the increasing cognitive benefits of code-switching among multilingual individuals (Adesope et al., 2010) provide strong evidence that there is a need to facilitate easy switching between languages. It is common to hear bilinguals naturally switch between two languages in the same sentence during a conversation. However, the same level of convenience is not reflected while typing because the current text input mechanism (see Fig. 3.1) is not optimal yet. Furthermore, Emoji has also become a universal language that the millennial demographics (ages 18 to 34) rely on to communicate their thoughts beyond mere words (Steinmetz, 2017). Hence, there is room for improvement where interaction with mode can play a significant role in supporting the diverse scenarios of typing, potentially shaping our word preferences and cognitive impacts.
3.2 Approach

Our overall methodology involved first modelling the existing interaction, and then inefficiencies were identified and addressed by proposing enhanced features. Subsequently, the proposed models were evaluated by comparing the number of taps required to complete typical typing tasks.

A previous study (Degani, 1996) showed that Statecharts (Harel, 1987) could be used to model moded interaction in complex aeroplane control systems effectively. Hence, we consistently applied this modelling language to our selected context of the multitouch smartphone system to better understand the transitions and relationship between modes of a soft keyboard. The specific system in which the modelling and analysis were conducted was iPhone X iOS 12. However, we also addressed how our approach could also make sense for designs deployed in alternative ecosystems, including Samsung Galaxy S9’s Android 8.0 and Lumia 950’s Windows 10. Due to the hierarchical design observed, the keyboard-typing model can be further decomposed into 2 layers of mode-switching interaction: between languages and within a language. Each layer will be elaborated separately so that a comparison between existing and proposed models of interaction within each layer can be made more clearly. Our

\footnote{Since some languages like English and Bahasa Indonesia share the same Latin characters, the languages we call ‘modes’ in our study refer to those which visually differ in characters. Hence, we are aware and take into consideration the fact that Indonesian typists can still type using the English (US) keyboard mode}
propose models will revolve around re-structuring the mode arrangement and incorporating UM technique.

### 3.2.1 Layer 1: Between Languages

Referring to Fig. 3.1, let us assume that the user is a bilingual who uses both English and Korean languages daily (or any non QWERTY layout). Together with Emoji as a universal language (Hussey, 2015), there are a total of 3 modes available on his/her keyboard. If more languages were to be added manually in reality, the number of modes would increase accordingly. Following the above assumption, a black box represents a modes, while an arc represents a directional transition between one mode and another (see Fig. 3.2). Each arc is accompanied by a label in the format of ‘A(B)C’ which describes a mode-switching technique B, manifested by an interaction gesture C on interface element A. The small grey node D points to the default mode of the system. This same representation strategy was consistently applied to the remaining system models constructed throughout our study.

**Figure 3.2:** Existing model of mode-switching between languages

**Figure 3.3:** Instead of looping through intermediate modes, one can hold (left) on globe icon and tap (right) on a desired mode
Chapter 3. Optimising Moded Interaction Design for Soft Keyboard

Figure 3.4: Proposed UI design for mode-switching between languages. Changes were illustrated using red boxes.

Existing Model

From Fig. 3.2, it can be observed that there exists a single directional cyclic arrangement between the 3 modes. While the order of this cycle can be customised in the keyboard settings by the user at any time, once confirmed, the convenience of single-tap access will only persist each mode in that static order. This means a single tap does not allow users to jump from ‘Latin’ (QWERTY) to ‘Korean’ mode instantly, without having to go through intermediate modes. However, there is an option to break away from this single directional loop by holding onto the globe icon and then a single tap on the target language (mode) to switch to (see Fig. 3.3). On a Windows phone, the same moded loop also exists between languages, but the difference is that it can be triggered by swiping the spacebar left or right, making it a bidirectional loop instead.

Proposed UI

In an Android or versions older than iOS 12, the Emoji mode can be accessed from any other mode through a dedicated button. This should be applied consistently to all the available modes at the same level. Using the available space at the bottom of the keyboard (see Fig. 3.4), we introduced new buttons that can be single-tapped to persist any mode from any other mode. This means that the globe icon can be substituted with a settings icon so that the language selection will no longer be nested and hidden anymore. We chose a darker shade of grey to represent the mode keys to differentiate from the within-language keys while simultaneously maintaining the overall design consistency. Furthermore, the currently active mode key will have contrasting visual transparency to show that its functionality has been applied and hence, tapping it one more time does not apply any change to the system.

Proposed Model

The change is also reflected in the proposed model (see Fig. 3.5). We ensured that there always exists a yellow bidirectional arc to flexibly switch between any two modes by only a single tap, instead of the current inefficient hold-and-tap mechanism. Furthermore, since the frequency of keys pressed in Emoji and second language (which is Korean in this assumption) mode is relatively lower than that of the default Latin alphabets, persisting a mode temporarily may not be the most efficient way to go. The
unique strength of UM can be leveraged here because while holding a mode button temporarily, users will be able to type the less frequent language and instantly return to the previous language upon releasing the mode button. This means that a single tap can now chunk the cost of switching from one mode to another and vice versa. Therefore, we incorporated two pairs of UM’s hold-and-release arcs, pointing to two different directions between any two languages. Each pair has a blue line to indicate a hold action and a red dotted line to indicate a release action, as shown in Fig. 3.5. The ability to trigger UM can now be accessed from any language mode in the system. Overall, this proposed model incorporated a dual functionality of Persist and UM technique on every transition between any 2 modes.

3.2.2 Layer 2: Within a Language

Existing UI

Even if the user is monolingual and does not switch between any language, moded interaction also exists within a single language. For instance, while in a default Latin/QWERTY mode, we can identify 4 available sets of keys (see Fig. 3.6) that support a fully functional keyboard typing: uppercase, lowercase, number with punctuation (numPunct) and symbol with punctuation (symPunct). These sets can be considered modes because each set’s layout design is preserved while the different labels are printed in each key/button to represent the different sets. This consistent layout may have minimised the contextual differences between each mode, but prior to pressing each key, the user...
is made aware of the currently active mode through the different printed labels. Hence, there is no mismatch between the mental model of the user and that of the system.

**Existing Model**

In iOS 12 and older versions, the default mode is of lowercase characters and assuming user preferences like auto-capitalisation were toggled off, we draw the corresponding Statecharts (see Fig. 3.7). The key can be double-tapped from the lowercase mode to persist the new mode of uppercase characters or single-tapped to trigger an automatic release back to the lowercase mode immediately upon the tapping of subsequent keys. Once technique is suitable because of the natural expectation that only the first letter of a word would typically be capitalised to follow grammatical rules. In cases like acronyms, where users need to type the uppercase character more than once, the key can alternatively be maintained using UM technique to enjoy the benefit of automatic release. As shown in Fig. 3.7, the numPunct mode can also be directly accessed from the default lowercase mode while the symPunct mode cannot. Instead, symPunct mode can only be directly accessed from numPunct mode. This nesting of modes only increases the complexity of moded interaction. However, there is a shortcut in the system design, allowing both numPunct and symPunct modes to have a quick return to the default lowercase mode by single-tapping the spacebar which simultaneously inserts a blank space.

**Proposed Model**

Overall, the lack of consistency in the provided techniques and transitions should be addressed to facilitate a versatile mode-switching experience. For instance, we proposed to equip the existing model with additional two UM transitions: from uppercase to lowercase and from symPunct to numPunct. On top of that, the linear structure of nested modes can be redesigned to reflect a flat hierarchy of modes. Each mode has equal access directly to any other mode in the system without having to go through any intermediate modes. Firstly, we grouped the lowercase and uppercase modes and the transitions between them into one sub-system (see Fig. 3.8). Then, treating this group as a discrete mode, we can now minimise the visualisation of the system to just 3 modes: lowercase/uppercase, numPunct and symPunct. Similar to Fig. 3.5, we ensured a pair of bidirectional Persist and UM techniques in between any 2 modes.
Figure 3.7: Existing model of mode-switching interaction within Latin/QWERTY mode.
Figure 3.8: Proposed model of mode-switching interaction within Latin/QWERTY mode.
Chapter 3. Optimising Moded Interaction Design for Soft Keyboard

Proposed UI

As shown in Fig. 3.9, the space dedicated for the initial 123 and → buttons has now been divided to form 4 new distinct buttons. We ensured that numPunct and symPunct modes could be equally accessed from both uppercase and lowercase mode, instead of having symPunct nested within numPunct. For instance, tapping the 123 button will persist into numPunct mode while tapping #= button will persist into symPunct mode. Triggering the [abc] button from numPunct/symPunct mode will return the user to the default lowercase mode. Since there is no corresponding uppercase effect for numbers, symbols, and punctuation characters, the ‘Shift’ button in both numPunct and symPunct mode can be replaced with ‘ABC’ label to facilitate direct access to uppercase mode, without having to go through lowercase mode first.

Inception of SoftCuts

Despite reducing the size of mode keys, they are still relatively bigger than that of individual character key. We also ensured that the proposed user interface design (see Fig. 3.9) still preserves its original design layout of vertical symmetry. Such a strategy gives us the opportunity and space to incorporate one new function into the proposed keyboard. After observing how efficient the modifier keys like (command) or ctrl has been for our interaction on a tangible keyboard, we wondered why our digital keyboards lack such features. Shortcuts like copy ⌘+C, paste ⌘+V, undo ⌘+Z, bold ⌘+B, italicise ⌘+I, and underline ⌘+U are becoming common knowledge across different OS and especially for expert users. The difference is whether one is using a PC (ctrl key) or MacOS keyboard (⌘ key). Currently, styling actions like bold, italic and underline can only be accessed through direct manipulation on the toolbar of a smartphone while an undo action will only be triggered upon shaking in the case of the iPhone. By leveraging previous knowledge from using a laptop/desktop keyboard, we can facilitate the access of visually hidden features of an application. This is especially true when it is not optimal to crowd the already limited screen real estate with more representation of the different functions.

In current keyboard designs, there exists minor visual inaccuracy in two transitions: from numPunct to lowercase mode, and from symPunct to lowercase mode. Instead of the ABC label, it would be more accurate to have abc printed instead because the transition is not from numPunct/symPunct to uppercase mode, but from numPunct/symPunct to lowercase mode instead. Similarly, the ambiguous ↑ label
in uppercase and lowercase mode should also be substituted with \texttt{abc} and \texttt{ABC} label respectively to describe the transition more accurately, instead of the current ‘up’ arrow (see Fig. 3.6). Such corrections had been reflected in the proposed user interface design (see Fig. 3.9).

### 3.3 Quantitative Evaluation

After having both existing and proposed models of interaction elaborated in the previous section, we now validate the improvements in terms of theoretical calculation. The number of taps needed to type a string of characters can objectively represent the time needed and complexity of the interaction. Hence, we will show different examples of strings that involve multiple layers of moded interaction, then compare the number of taps needed by existing and proposed models.

#### 3.3.1 Layer 1 Comparison: Between languages

Bilinguals often switch between languages to highlight the cultural or contextual meaning portrayed by each language. Fig. 3.10 shows an example of a typical sentence in a text messaging app that switches between Latin alphabets, Emoji characters, and Korean alphabets. In the existing system (see Fig. 3.2), we count the mode-switching triggered by hold-and-tap as double of that triggered by a single-tap to account for the greater effort and time needed. Since its functionality is nested inside the globe icon (see Fig. 3.3), it also results in poor discoverability among novice typists. Therefore, we used the more obvious mode-switching strategy to calculate the total number of taps needed by the existing system. However, by leveraging UM’s chunking effect, the proposed system can now avoid repetitive mode-switching taps and significantly reduce its total number to one-third its initial, \textit{i.e.} from 9 taps in the existing system to only 3 in the UM-only system (see Fig. 3.10).
Chapter 3. Optimising Moded Interaction Design for Soft Keyboard

FIGURE 3.11: Evaluation of strategies switching modes within a language to type a password or hashtag: #CH!UX!D2019. The password is inspired by the conference name ‘CHIuXiD’ where we presented this chapter of this thesis.

<table>
<thead>
<tr>
<th>Mode-switching pattern for string: #CH!UX!D2019</th>
<th>Number of mode-switching taps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Existing System (Persist only)</td>
<td>11</td>
</tr>
<tr>
<td>Proposed System (Persist only)</td>
<td>7</td>
</tr>
<tr>
<td>Proposed System (Persist + UMI)</td>
<td>5</td>
</tr>
</tbody>
</table>

FIGURE 3.12: Evaluation of strategies switching modes within a language to type a mathematical expression.

<table>
<thead>
<tr>
<th>Mode-switching pattern for string: [A - b] % C * d</th>
<th>Number of mode-switching taps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Existing System (Persist only)</td>
<td>13</td>
</tr>
<tr>
<td>Proposed System (Persist only)</td>
<td>8</td>
</tr>
<tr>
<td>Proposed System (Persist + UMI)</td>
<td>7</td>
</tr>
</tbody>
</table>
Chapter 3. Optimising Moded Interaction Design for Soft Keyboard

3.3.2 Layer 2 Comparison: Within a Language

There also exist real-life scenarios where the current keyboard design may not be optimal for complex sets of strings that involve multiple mode-switching within the same language. For instance, password input is often mandated to involve different categories of characters to minimise the susceptibility to password cracking (Summers and Bosworth, 2004). Fig. 3.11 shows the evaluation of a possible manifestation of the complex yet easy-to-remember password. Another example would be entering a mathematical formula, which is commonly regarded as quite inconvenient in typing in a multitouch platform. Both Fig. 3.11 and 3.12 show that our proposed system managed to half the total number of mode-switching taps, i.e. from 11 to 5 in the former and from 13 to 7 in the latter.

Our overall strategy was to first flatten the hierarchy in the existing structure of modes, such that one mode is always one tap away from any other mode in the system. This reduces the need to nest one mode inside another, which only increases the complexity of the interaction. Secondly, we equipped each transition between any two modes to adopt a dual functionality of both Persist and UM techniques to provide the most versatile mode-switching interaction with a virtual keyboard on a multitouch platform can offer. We avoided the situation where we need to choose either technique and sacrificed the advantage of the non-chosen one. By implementing both, the users’ mental model will always be correct regardless of whether one chooses to trigger the traditional Persist or the efficient UM. Lastly, in terms of user interface design, we ensured that the correct label is to be rendered on every mode button to maintain the consistency and accuracy of the experience.

3.4 Discussion

Our proposed system will have a varying extent of improvement depending on the pattern of interaction. As shown in Fig. 3.13, the flat hierarchy design ensured no more

<table>
<thead>
<tr>
<th>MODE-SWITCHING PATTERN FOR STRING: while(var1==x){return true}</th>
<th>NUMBER OF MODE-SWITCHING TAPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Existing System (Persist + UMI)</td>
<td>9</td>
</tr>
<tr>
<td>Proposed System (Persist only)</td>
<td>8</td>
</tr>
<tr>
<td>Proposed System (Persist + UMI)</td>
<td>7</td>
</tr>
</tbody>
</table>

Figure 3.13: Evaluation of strategies switching modes within a language to type in a programming syntax.
than two consecutive mode-switch triggers, hence reducing the number of taps from 9 to 8, not as much as the previous examples do. We can further exploit opportunities presented in an alternating pattern like A-B-A, where A represents one mode and B represents another. Although it only reduced the number of taps by one, such an ideal pattern can be optimised by incorporating UM to prevent additional taps from being triggered and wasted. In other words, the number of taps saved by employing the proposed mode re-arrangement depends on the patterns of the text to be typed in. One can imagine other example sentences where there is no difference in the number of taps at all. Therefore, such insights help inform which scenarios and mode-switching patterns will be optimal to leverage the potential of the proposed strategy.

While the number of taps is a sufficient yardstick of comparison in terms of speed and accuracy of interaction in each system, a future phase of this study should include usability testing with both novice/expert typists and a longer set of complex strings. This is so that more statistically significant differences can be observed and learned from in terms of actual usage and user experience while taking into account factors like learning and familiarity of users. New challenges may also arise in attempting to bridge the theoretical value to the commercial level. For instance, since the UM technique requires holding a widget to maintain a given mode, it may handicap the typist from exploiting the full benefits of 2-finger typing. Therefore, alternative input modality can be explored and developed to support the UM technique of mode-switching in the future.

Regarding scalability, we are aware that the space at the bottom of each screenshot (see Fig. 3.4) can only afford a limited number of layouts needed by the different languages. Although our study only showed mode-switching between Latin, Emoji and Korean keyboard layouts, future studies could explore alternative design strategies to address scenarios where the typist knows more than 4 languages.

Considering how considerable the inertia can be in changing users’ habit of typing on a smartphone, this study can be perceived to have limited impacts. However, the primary objective of this study is not to prove a better typing experience but more of providing an alternative perspective through mode to better understand the fundamental interaction we take for granted. Therefore, no major changes to the overall layout and the current typing practice were introduced as we intended to preserve the system’s original design. The improvements in theoretical calculation further validate that our study is in the right direction at least from the efficiency (in the number of taps required) point of view when the phrases or sentences to be typed form a particular pattern. More work can be done to leverage this efficiency with the patterns to maximise the impact.

3.5 Conclusion

By modelling the existing moded interaction in multitouch typing using Statecharts, we could identify missing transitions and inefficient mode hierarchy between languages and within a language, as equipped in most commercial multitouch platforms today.
Then, we proposed alterations to both the interaction model and user interface design and further validated it by comparing the number of mode-switching taps needed in different systems. This way, we could show how the often overlooked and simplistic way of typing on our smartphone can be enhanced to support diverse contexts where more than 2 modes are involved. Beyond typing, instances like web browsing, map navigation, and article reading characterise behaviours where one would need to constantly switch between tabs, locations, and sections respectively, to compare elements or perform multi-tasking. For example, while reading an academic paper, readers often have to temporarily jump to the reference section to make sense of the in-text citation before reading the main section.

In particular, this Chapter has opened up opportunities for a soft keyboard to support command-selection mode (i.e. SoftCuts). The new proposed (see Fig. 3.9) motivates us to explore how the keyboard designs would visually transform to feature the available commands, which we will discuss in the following Chapters.
Chapter 4

Maximising Discoverability of SoftCuts

4.1 Motivation

To use SoftCuts, users must first discover two of its essential visual elements: the modifier key and the command items to be chorded together. In this chapter, we focus on investigating how both of them could be designed to maximise the discoverability of SoftCuts, through two studies. The first study evaluated the design rationale and preference between three visual representations for the on-screen command keys, showing either the letter only (D1), the letter and command name (D2), or the command name and an icon (D3), through semi-structured interviews and subjective ratings. The second study investigated two scenarios with two different design factors for the modifier keys. We varied the saliency of modifier keys using one scenario (i.e. note-taking) where the keyboard was always displayed on the screen and another scenario (i.e. web-browsing) where the keyboard was hidden by default. We also varied the familiarity of modifier keys using either $\text{⌘}/\text{ctrl}$ (familiar to Apple/Windows/Linux users) or $\mathcal{F}$ (a completely new and arbitrary design).

4.2 Study 1: Visual Representation of Commands

There is no broad consensus on how to present the hotkeys on a soft-keyboard. From Fig. 1.4, we can see that Samsung (Samsung, n.d.) highlights the key associated with command while Microsoft (PandaSage1221, 2013) provides the name of the command in addition to the letter name. Alternatively, from Fig. 2.10, we can see how keyboard layouts could also render command icon labels in each key. Indeed, the dynamic nature of soft keyboards being rendered on screen raises the question of how SoftCuts could be best presented to the user. We thus first decided to investigate these variations in visual representations and understand which one is preferred by users. We believe that only showing the letter will make it hard for first-time users to guess what the technique does and make it hard for users to find a command they want to use. However, it is unclear which of the command name or icon-based designs should be recommended. To address the lack of clarity, we conducted semi-structured interviews and subjective ratings to evaluate the usability of 3 design variants (Fig. 4.1).
Table 4.1: Design space for the visual representation, using Letter (key name), Name (of command) and Icons.

<table>
<thead>
<tr>
<th>Letter</th>
<th>Name</th>
<th>Icon</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>D2</td>
<td>X1</td>
</tr>
<tr>
<td>D2</td>
<td>X2</td>
<td>D3</td>
</tr>
<tr>
<td>X1</td>
<td>D3</td>
<td>X3</td>
</tr>
</tbody>
</table>

Figure 4.1: Design variations for SoftCuts. (a) We modified the existing iOS keyboard by adding two command keys at the bottom row. (b) Design 1: Available shortcut keys retain their default appearance, while non-shortcut keys are greyed and inactive. (c) Design 2: In addition to Design 1, the name of each command is printed at the bottom of respective keys. Design 3: Similar to Design 2, but substituting the key labels with icons instead.

4.2.1 Design Variants

Table 4.1 enumerates all possible combination between relevant representations (letter, icon, and name) of a command/key. We included half of them (D1, D2, and D3) for further evaluation and intentionally excluded the other half (named X1, X2, and X3) to avoid a long study duration. They are not practical to be implemented.

D1: Letter only.

As illustrated in Figure 4.1b, this design deactivates keys that do not provide any shortcut functionality and focus the users’ attention on active keys that offer shortcut functionality. D1 corresponds to the one currently being implemented on a Samsung tablet (Samsung, n.d.). This design has the least amount of change from a standard soft keyboard among all the designs. It requires users to have prior knowledge of the available commands because the system does not provide mapping information between keys and commands. Novice users would be at a disadvantage because they would first have to guess the associated function and learn through trial and error or attempt to transfer skill learned from a different platform, resulting in a potential mapping mismatch.

D2: Letter and name.

In this design, the command name is added at the bottom of each key, right below its corresponding letter. D2 is similar to both KeyMap (Lewis, d’Eon, et al., 2020) and the one currently implemented on Microsoft Surface tablets (PandaSage1221, 2013) while
only differing in command name position and saliency (Figure 4.1c). This design facilitates direct mappings between each letter and its corresponding command output. Over repeated triggers, D2 may help long-term retention of explicit mapping printed on each key and potentially support skill transfer from soft keyboards to physical keyboards.

D3: Icon and name.

Figure 4.1d shows how this design hides the letter from the key to only show the command name and a corresponding pictorial representation (icon). D3 has only been used by some physical keyboards (Block, Gellersen, and Villar, 2010; Studio, 2011). This design uses icons as a visual cue for guiding visual search (Bailly, Lecolinet, and Nigay, 2016) and conveying command meaning (Giannisakis et al., 2017).

Discarded Designs

Icon and Letter (X1). While Giannisakis et al. (2017) proposed various ways how we could blend keyboard shortcut information (letter) into toolbar buttons (an icon within a key), it is challenging to apply such a strategy consistently throughout all commands. For instance, commands like ‘bold’, ‘underline’, and ‘italicise’ share the same representation between icon and letter due to mnemonic mapping. If we were to proceed with the icon-and-letter configuration, each key would end up displaying a similar-looking ‘B’ twice, the first time to represent the $\text{ctrl} + \text{B}$ hotkey information and the second time to represent the established icon (e.g. bolded ‘B’).

Command Name only (X2). To display a command name on a small square key, we need to use significant horizontal space. As such, the font size to display the command name will be limited, and there will always be extra space above and/or below the command name. Instead of leaving the space empty, we decided to either display the Letter or an Icon to provide more information.

Icon only (X3). Icons can help visual search very efficiently, but not every command has an associated icon. In other cases, participants may not recognise the icon or map it with an existing command. We thus saw this condition as impractical for users. A similar issue happens with D1 (Letter only – no information about the command), but we still included it because it is already implemented on commercial products (Samsung Galaxy Tab).

4.2.2 Procedure

We conducted the study through a video conferencing platform, where we started with participants digitally signing a consent form and then downloading the prototypes on their phone. The prototype offered two simulated apps: “Notes” for note-taking and “Safari” for web-browsing tasks. For the first app, we instructed participants to complete tasks using one of the design variants before they rated their experience in terms of preference, ease of use, perceived speed, perceived accuracy, usefulness, comfort, and visual appeal using a 7-point Likert Scale. Also, participants were invited to an
Figure 4.2: A demonstration of tasks in a note-taking app that may be done using existing mechanisms or SoftCuts’ invocations to complete. Participants had to paste text, (a) select all the pasted text, (b) bold the selected text, (c) undo the bold formatting then select all the text again, and finally (d) colour the selected text red. Step (c) and (d) could only be achieved through SoftCuts.

optional open-ended discussion to suggest any feedback and ask the experimenter any question. They would repeat the tasks, rating, and discussion phase with the second and third design variant. We then finally asked for their overall design preference (D1, D2, or D3) for that particular app. Before proceeding to the second app, the experimenter would then only finally explain the concept to each participant to simulate an authentic discovery of the SoftCuts for the first three conditions (in the first app). By the second (final) app, participants were expected to be familiar with the rationale behind SoftCuts already. We wanted to see how their comments would change as we switched the app in which they had to complete similar tasks. The session ended with asking participants which keyboard shortcuts they were most familiar with. For the full details of the questions we asked, please refer to Appendix A.

4.2.3 Task

We slightly altered the three design variants to accommodate the limited functionality of both the Notes and Browser app prototype. As seen in Figure 4.2b, 4.2c, and 4.3b, the white keys contain either black or light grey font. The black font represents functional commands, while the light grey represents disabled commands.

Note-taking App (Keyboard Task)

This app offers 16 shortcuts and features a scenario with a visual on-screen keyboard (Figure 4.2). We asked participants to assume that text had been previously copied and
stored in the OS clipboard. Then participants had to perform the following:

1. paste text,
2. select all the pasted text,
3. make it bold,
4. undo the bold command,
5. select the entire text again,
6. colour the text in red.

Commands 1, 2, 3, and 5 could be invoked either using the default menus or via SoftCuts. Commands 4 and 6 could only be invoked using the corresponding SoftCuts located in the Z and R keys. Keys for SoftCuts were chosen either because they correspond to the actual keyboard shortcut on the desktop for this command (e.g., V, A, Z, F, D) or because it was the first letter of the command name (e.g., R for red colour). We intentionally chose to rely on commands that could only be activated via SoftCuts to encourage participants to explore beyond their familiar menu-based approach and thus reveal the tested SoftCuts' visual design as they would when interacting with their device.

**Web-browsing App (Keyboardless Task)**

This app offered nine shortcuts keys and featured a scenario without any keyboard initially, with only the (modifier) key presented (Figure 4.3). When using this app, participants had to activate the following:

1. activate the "search on page" command and look for a given string,
2. activate the bookmark command.

These two functions could only be activated via SoftCuts, and no native approach was implemented. We chose the F key for find and the D key for bookmark function as we wanted to preserve the mapping that already exists on desktop computers.

### 4.2.4 Participants and Apparatus

We recruited 12 participants (10 female, 2 male and all right-handed), aged 20 to 23 (M=21.9, SD=1.0) from the university community. The recruitment was limited to iPhone users to ensure familiarity with iOS user interface design and the key. The majority of the participants reported that they used keyboard shortcuts either often or always on a range from 1 ("never") to 3 ("sometimes") to 5 ("always") (M=4.1, SD=1.0). We designed our interactive note-taking and web-browsing app prototypes using Figma.

---

2Note that the undo command on iOS can also be triggered by shaking the device, which we did not reproduce. Hence, it could be achieved via SoftCuts only.

3https://www.figma.com/
and hosted them via Maze\(^4\). Participants used their iPhones (models include iPhone X, XR, XS and 11) to run the prototype and a computer with access to WiFi to facilitate the online interview and subjective rating. Each participant received a S$7.50 reimbursement for their 45 minutes participation.

### 4.2.5 Design

We used a within-subject design with DESIGN as a primary independent variable and APP as a second independent variable. DESIGN has 3 levels \{ D1, D2, and D3 \} while APP has 2 levels \{ NOTES and BROWSER \}. The order of presentation of the two apps and three design variants were counterbalanced using Latin Square to avoid any potential ordering effect. In terms of dependent variables, we measured subjective feedback using a 1-7 Likert (1: strongly disagree, 7: strongly agree) on their overall opinion between design variant D1, D2 and D3 in terms of like, ease of use, speed, accuracy, comfort, usefulness, and visual appeal. The experiment lasted for around 45 to 60 minutes. In summary, we recorded 12 participants \(\times\) 2 apps \(\times\) 3 designs \(\times\) 7 rating questions = 504 ratings in total.

\(^4\)https://maze.design/
4.2.6 Results

For quantitative data analysis from Likert scale rating, we used Friedman tests (non-parametric) and Wilcoxon signed-rank tests with Bonferroni corrections for post hoc comparisons.

Notes (Keyboard Scenario)

The majority of participants liked D3, found it easier and more accurate to use than D1 in NOTES app (Figure 4.4). There were significant main effects of DESIGN on like (χ²(2, N = 12) = 8.49, p = .014), ease of use (χ²(2, N = 12) = 8.60, p = .014), accuracy (χ²(2, N = 12) = 10.61, p < .01), comfort (χ²(2, N = 12) = 7.82, p = .020), and appeal (χ²(2, N = 12) = 8.16, p = .017). Pairwise comparisons revealed that participants liked D3 (median=6.5) more than D1 (median=4.0, p = .044), perceived D3 (median=7.0) to be easier to use than D1 (median=3.0, p = .027), and perceived D3 (median=6.5) to be more accurate than D1 (median=5.0, p = .032). Scores for D1 tended to be lower than D3 but we did not find any significant differences between D1 and D2 or between D2 and D3 either.

Browser (Keyboardless Scenario)

Participants found D2 more comfortable to use than D1 in BROWSER app (Figure 4.5). There were significant main effects of DESIGN on speed (χ²(2, N = 12) = 6.93, p = .031), accuracy (χ²(2, N = 12) = 6.06, p = .048), comfort (χ²(2, N = 12) = 8.19, p = .017), and appeal (χ²(2, N = 12) = 7.17, p = .028). Pairwise comparisons revealed that participants found D2 (median=6.0) more comfortable to use than D1 (median=5.0, p = .039). There were no other differences between individual designs.

4.2.7 Discussion

Strong dislike for D1

Our initial guess is supported: users did not like the simpler design with the letter name only. None of the 12 participants chose D1 as their overall preferred design for
NOTES app, while only one chose D1 (P5) for BROWSER app. Note that P5 rated her hotkey usage frequency at 5 (i.e. all the time), which suggests that expertise may affect design preference. Our Likert data analysis also revealed that D1 was less favoured than D2 (in terms of comfort) and D3 (in terms of likeability, ease of use, and accuracy). Participants explained that “it is harder to know straight away that to italicise is to press $I$ and to cut is to press $X$” [P5], hence D1 “is less intuitive because one has to think of what the letters could possibly mean” [P10]. We also observed that 3 (P1, 7, 8) out of 4 participants, who had to use D1 as the first design variant, struggled to discover SoftCuts and required assistance to complete the instructed task. This is further supported by comments like “if I had not been exposed to the other designs [D2 and D3], this design [D1] would have been completely unintelligible to me” [P11] (similar with P12).

In addition, D1 also lacks feedforward and feedback cues. Participants found that the command name labels in D2 and D3 “make it easy to understand what is going to happen when you press the keys” [P1]. These labels also help to saliently distinguish the keyboards in hotkey mode from that in the default text-entry mode because without them, like in the case of D1, users may “think that they were still typing when in fact they were actually editing styles” [P3] (similar opinion for P6, 10), hence resulting in mode errors. One participant also commented, “I guess I will get used to D1 over time, but it will be troubling at the start, and I am not interested in such a design that makes my life harder” [P5].

Hence, the above observations suggest that D1 deserves the least consideration when implementing SoftCuts due to its steep learning curve, even for users who regularly use keyboard shortcuts on desktops.

Preference for D3

From our quantitative results in the Notes app, D3 was preferred over D1 in terms of likeability ($\Delta_{\text{median}} = 2.5$), ease of use ($\Delta_{\text{median}} = 4.0$), and accuracy ($\Delta_{\text{median}} = 1.5$). At least 75% of participants overall chose D3 over D2 for both NOTES (9 vs. 3) and BROWSER (10 vs. 1) app. While our data did not reveal any significant differences between D2 and D3, our qualitative data suggests that D3 was preferred. Participants explained that “it is easier and faster to infer what each key represents from icons as compared
to reading the name labels at the bottom” [P1] (similar opinions from 7 other participants). This is especially true when the icons are consistent with those used in existing commercial apps like Microsoft Word that they are already familiar with (mentioned by P3 and P5). Therefore, we would recommend D3 as a suitable design for SoftCuts.

D2 as the next best alternative

Despite a preference for D3, we were curious as to why some participants would choose D2 for Notes (3 persons) and Browser (1 person) app. One reason mentioned was that with the already limited screen space available for each key, letters in D2 on average consume less space than the icons in D3, resulting in “a cleaner, minimalist” [P2] and “less cluttered” [P10] presentation of commands in D2. Another reason could be that icons can be misinterpreted (Bailly, Lecolinet, and Nigay, 2016), especially with commands that do not have a standardised visual representation. Using both letter and command name on each key, D2 can build stronger shortcut mappings than D3 when new contexts may involve keyboards that do not offer any cues (e.g. physical keyboard or Samsung tablet’s soft keyboard (Samsung, n.d.)). Therefore, it is a worthy consideration to offer individual user customisation where they can “flexibly change preferred design between D2 and D3” [P6,10,12], depending on their level of familiarity.

Context familiarity affects preference

Several participants mentioned that they would prefer D2 on Notes but D3 on Browser (P2,3,7,8) because they were more familiar with keyboard shortcuts for text-editing commands than for web-browsing (P3,7). These claims can be supported by our Likert data, which shows that D3 received only 3 neutral (scores of 4) ratings, and the rest were all positively (scores of 5 or above) rated across the different questions asked during the Notes app. D3 received negative feedback in the Browser app, with 2 scores of 1, 2 scores of 2 and 8 scores of 3 overall. Beyond design comparison, the value of SoftCuts as a command selection mechanism seems to be appreciated more when participants have prior knowledge of keyboard shortcuts. For instance, both P7 and P9 had to use the Browser app first before using the Notes app. Both mentioned, “I finally feel SoftCuts is intuitive after trying out D3 on Notes app because it is similar to my computer usage” [P7,9], while one of them said, “I am not really sure if I need this [SoftCuts] for a mobile browser because I usually search things only in my computer” [P7]. This means that some participants could have been conditioned with their pre-existing limitations and hence were unaware that such an essential function like searching for text, could and should be made accessible despite the platform they were using it on. Hence, we postulate that only with greater integration and consistency across diverse applications and platforms will SoftCuts’ advantages be realised even more.

SoftCuts may facilitate knowledge transfer and expansion

Our assumption on leveraging prior knowledge is partially supported when participants affirmed: “it is pretty cool that [SoftCuts] mirror what you do on a physical keyboard” [P1]. This suggests a minimal performance dip (Cockburn, Gutwin, et al., 2014), “as long as one has been using physical keyboard shortcuts for a long time, it would be very
easy and not take too long to get used to SoftCuts on mobile devices” [P5]. Beyond intermodal performance improvement, SoftCuts also facilitate command browsing because “it lets me know what other commands are available for future invocations” [P9]. As such, it may increase participants’ mapping vocabulary through implicit learning and frequent exposure. For instance, despite not being tasked to invoke a redo command, P7 expressed his discovery through SoftCuts that $[\text{Alt} + \text{Y}]$ shortcut can be used to trigger the redo command next time. Therefore, these observations suggest how robust SoftCuts could be in supporting and even growing expert performance.

Potential enhancements for SoftCuts

One of the suggestions shared by the participants was to “merge D2 and D3 together to have all three types of cues (command name, letter and icon) in one key” [P2]. This is possible if we adopt KeyMap (Lewis, d’Eon, et al., 2020)’s design by augmenting our D3 design with a small letter displayed at the top right corner of each key by adding a small letter to the top right corner of each key in our D3 design. However, due to the space constraint on a phone, it may be more practical to restrict this hybrid design to tablet devices. Another suggestion was to adopt “a transparent modifier key” [P2,5,7] when used without a full keyboard so that it does not occlude content while reading in a browser app. To accommodate requests to “include new commands like commenting, highlighting or translating texts” [P3,10], participants suggested “individual customisation where they could remap their preferred keyboard shortcut mapping to suit their productivity workflow” [P6,10]. It is thus worthwhile to consider these suggestions for future implementations and evaluations of SoftCuts.

4.2.8 Limitations

Design combinations like D2, D3, and X2 face space constraints when accommodating long command names like “Paste and Match Style”. Decreasing the font to fit the tiny keys on a smartphone will only worsen the accessibility as the current font size may have already posed a challenge for an older population. However, visualisation strategies could be explored in future studies. For instance, when users are holding a particular key with a finger, a bubble callout (Vogel and Baudisch, 2007) can be used to magnify the small font or offset the presentation of long command names to the available space above the keyboard. Another strategy is to adopt scrolling animation. Alternatively, designers could also establish a standardised icon to represent these commands such that the icon alone is sufficient to communicate the function of the key. We also acknowledge that our study with participants of 20-23 years old who have used hotkeys frequently may be limited to generalise the results. Additional studies are needed to confirm if a similar design preference could be observed for novices or someone who has not used hotkeys at all.

4.3 Study 2: Familiarity and Saliency of Modifier Keys

The concept of SoftCuts postulated that users’ familiarity with desktop keyboard shortcuts would make it easy to discover, but there has been no prior work to confirm this
assumption. Therefore, our goal was to assess whether users spontaneously discover that they can use SoftCuts, that is, if users would start to use SoftCuts without being explicitly introduced to them prior. The discovery of an interaction technique is a binary measure. As such, it can only happen once for each participant, and repeated measures are impossible. Thus, it requires many participants to generate insightful analysis, which is why we chose to recruit participants from Amazon Mechanical Turk to perform a series of tasks in this experiment. Some tasks required users to select commands that could be activated either using SoftCuts or using the native menu/toolbar. Other tasks required users to select commands that could only be activated using SoftCuts, requiring participants to discover the technique.

We were interested in finding out if having the modifier key alone on screen would give a more explicit hint to participants and allow them to discover SoftCuts more easily. In keyboardless (noKB) conditions, participants would see a single modifier key on the bottom left of their screen (Figure 4.6). In keyboard (KB) conditions, participants would see the whole on-screen keyboard with the modifier key located at either bottom left or bottom right corner of the said keyboard. These conditions vary the saliency of the modifier key. We also considered two different visual designs for the character used on the modifier key: a familiar / label and a custom one . The input method used for SoftCuts in this study is the “Once” technique, where commands are selected by sequentially tapping on the modifier key and then on the command key. Based on the results of our previous Study 1 (see Section 4.2), we decided that when the modifier key is pressed, keys that could activate commands will display the icon and the name of the corresponding command (i.e. D3).

4.3.1 Missions and Tasks

We designed two missions for participants to complete, consisting of tasks replicated from the previous study using the Notes app prototype. Mission1 features three tasks: pasting, selecting, and bolding texts. A unique design of Mission1 is that all tasks can be completed using either the default menus or SoftCuts. Mission2 also features three tasks: undoing commands, selecting, and colouring texts. However, unlike Mission1, the first and last task are only doable using SoftCuts. As such, Mission2 can only be completed if the user has discovered SoftCuts. Each mission offers a 'Give up' button.
for participants who wish to end their tasks early, but we inform participants at the beginning of the study that their rewards are based on their performance.

4.3.2 Procedure

After participants signed the consent form and answered demographic questions, we assigned them to one of the four conditions, which are combinations of saliency (KB vs. noKB) and familiarity (Cmd/Ctrl vs. Ctrl) of modifier keys. The four conditions are KB+Cmd/Ctrl, KB+Custom, noKB+Cmd/Ctrl, and noKB+Custom. We asked participants which desktop OS they are most familiar with: macOS, Windows, or Linux. If they chose macOS, we would use the label in the modifier keys for both KB+Cmd/Ctrl and noKB+Cmd/Ctrl conditions. Else, we would use the Ctrl label instead. In each condition, participants had to pass Mission1 first before proceeding to Mission2. After completing both missions, participants were introduced to the concept (the what, the how, and the why) of SoftCuts. We then asked them to rate their experience with SoftCuts using a 7-point Likert scale. To test participants’ level of attention, we asked a trick question that instructs the participant to choose a given option intentionally. While our prototype was using the D3 design variant of SoftCuts, we asked participants which one they would prefer if compared with D1 and D2, as featured in our previous Study 1. To capture richer qualitative feedback, we had an open-ended section where participants could type any other thoughts or suggestions about SoftCuts. For the full details of the questions we asked, please refer to Appendix B.

4.3.3 Measuring Discoverability

For each mission, we count how many participants were able to discover SoftCuts under different conditions. We distinguish three primary metrics to measure discoverability.

Spontaneous discovery rate

This metric measures discovery of SoftCuts when users are not required to use it to complete a task. Similar to (Appert, Chapuis, and Pietriga, 2012; Goguey, Casiez, et al., 2018), it is computed as the ratio between the number of participants who used SoftCuts in Mission1 and the number of participants who started Mission1 (Eq. 4.1).

$$\text{Spontaneous discovery rate} = \frac{\text{# of participants who used SoftCuts in Mission1}}{\text{# of participants who started Mission1}}$$

(4.1)

Enforced discovery rate

This metric measures discovery of SoftCuts when users need it to complete a task. It is computed as the ratio between the number of participants who used SoftCuts only in Mission2 (but not in Mission1) and the number of participants who had not used SoftCuts in Mission1 and started Mission2. This is because 1) only upon completing Mission1 successfully will participants be allowed to proceed to Mission2 and 2) discovering SoftCuts in Mission1 means that participants already knew about SoftCuts.
Figure 4.7: A desktop version of the mobile prototype being deployed to each crowdworker.

when in Mission2 and did not need to ‘discover’ it again (Eq. 4.2).

Enforced discovery rate = \frac{\# \text{ of participants who used SoftCuts only in Mission2}}{\# \text{ of participants who started Mission2 but did not use SoftCuts in Mission1}}

(4.2)

Overall discovery rate

We measure the overall discovery rate as the ratio between the number of participants who used SoftCuts at least once during the study and the total number of participants (Eq. 4.3). This is not a simple addition between spontaneous discovery rate and enforced discovery rate because there may be duplicates when the former overlaps with the latter. Participants who use SoftCuts in Mission1 may use it again in Mission2.

Overall discovery rate = \frac{\# \text{ of participants who used SoftCuts during the study}}{\text{Total \# of participants in the study}}

(4.3)

4.3.4 Participants and Apparatus

A total of 248 participants were recruited from the United States using Amazon Mechanical Turk. 160 of them (77 were male, 81 were female, 1 of them non-binary, and 1 preferred not to disclose) passed the attention test and were assigned to one of the 4 conditions evenly. Participants age ranged between 19 and 75 years old (\(M=36.3, SD=12.2\)). We required participants to have at least 95% approval rate. We also granted qualifications to workers to ensure that they could only complete the experiment once. Given that our experimental interface was based on the iOS Notes application, we also required participants to be familiar with iPhones. For smartphone, only 2 (1.3%) of the participants had 0 to 2 years of experience, 15 (9.4%) had 2 to 5 years, 121 (75.6%) had 5 to 15 years, and 22 (13.8%) had more than 15 years. For computer, 10 (6.3%) of them
had 2 to 5 years of experience, 58 (36.3%) had 5 to 15 years, and 92 (57.5%) had more than 15 years. The majority reported that they used keyboard shortcuts either often or always on a range from 1 ("never") to 7 ("always") ($m=5.0$, $SD=1.8$). Giving up Mission1 results in reimbursement of US$0.26 for a maximum of 2 minutes of their participation. Giving up Mission2 results in reimbursement of US$0.65 for a maximum of 5 minutes of their participation. If participants did not give up at all, they would be reimbursed US$1.30 for a maximum of 10 minutes of their participation.

We deployed the same interactive note-taking app prototype (see Fig. 4.2) used in the previous Study 1. Participants had to access the experimental interface from a desktop or a laptop computer using any web browser they desired (see Fig. 4.7). Pointing and clicking interaction with the prototype can be facilitated using a mouse or a laptop’s trackpad. We excluded mobile device interactions in the study for two reasons. First, the screen size of a tablet is significantly larger than that of a smartphone. It is challenging to ensure that all participants use a suitable model of similar screen size. Second, we cannot control the fact that some participants may choose to use their mobile devices with an external keyboard, which supports keyboard shortcuts and potentially influences our results. In addition, we are primarily investigating behaviour differences instead of performance. Therefore, our strategy is to render the smartphone prototype on a desktop/laptop computer to maximise the quality of data collected while minimising variation in device ergonomics.

4.3.5 Design

We used a $2 \times 2$ between-subject design with two independent variables: Familiarity { CMD/CTRL, CUSTOM } and Salience { keyboard(KB), keyboardless(NOKB) }. We also have extraneous variables on users’ experience, namely hotkeys’ Usage frequency { 1-7 }, Quantity known/used { 1-5, 6-10, 11-20, 21-30, >30 }, and Usage duration { 0-2, 2-5, 5-15, >15 years } to better understand if users’ familiarity with hotkeys do affect our results. In terms of dependent variables, we measured three discovery rates (spontaneous discovery rate, enforced discovery rate, and overall discovery rate). We also measured subjective feedback using a 1-7 Likert Scale (1: strongly disagree, 7: strongly agree) on their opinion using SoftCuts in terms of usefulness, ease of use, discoverability, memorability and visual appeal.

4.3.6 Results

A total of 9 (5.6%) participants failed Mission1, 59 (36.9%) passed Mission1 but failed Mission2, 92 (57.5%) passed both Mission1 and Mission2. Out of those who completed both missions, 2 (2.2%) preferred D1, 14 (15.2%) preferred D2, while the remaining 76 (82.6%) preferred D3. We distinguished between Mission success rate and SoftCuts discovery rate in that the participant might have failed a given mission but still managed to activate a command using SoftCuts during the failed mission. Then, we analysed discovery rates using a Chi-squared test with necessary Yates’ continuity correction, followed by pairwise Chi-squared tests with Bonferroni corrections for post-hoc comparisons. For subjective rating, we used the Kruskal-Wallis test, followed by pairwise Mann-Whitney U tests with Bonferroni corrections for post hoc comparisons.
Chapter 4. Maximising Discoverability of SoftCuts

**Figure 4.8:** Spontaneous discovery rate of SoftCuts (a) across all conditions, (b) aggregated by FAMILIARITY (i.e. type of modifier key used), and (c) aggregated by SALIENCY (i.e. presence of keyboard).

**Figure 4.9:** Enforced discovery rate of SoftCuts (a) across all conditions, (b) aggregated by FAMILIARITY (i.e. type of modifier key used), and (c) aggregated by SALIENCY (i.e. presence of keyboard).

**Figure 4.10:** Overall discovery rate of SoftCuts (a) across all conditions, (b) aggregated by FAMILIARITY (i.e. type of modifier key used), and (c) aggregated by SALIENCY (i.e. presence of keyboard).
Chapter 4. Maximising Discoverability of SoftCuts

Figure 4.11: Spontaneous discovery rate of SoftCuts across (a) frequency of hotkey usage, (b) quantity of hotkey known/used, and (c) years of experience using hotkeys.

Figure 4.12: Enforced discovery rate of SoftCuts across (a) frequency of hotkey usage, (b) quantity of hotkey known/used, and (c) years of experience using hotkeys.

Figure 4.13: Overall discovery rate of SoftCuts across (a) frequency of hotkey usage, (b) quantity of hotkey known/used, and (c) years of experience using hotkeys.
Discovery Rates

Spontaneous discovery rate  Overall, participants discovered SoftCuts spontaneously at a similarly low rate of 18.1% (29 out of 160). There was no main effect of Familiarity or Saliency on spontaneous discovery rate ($\chi^2 = 1.39, p = .7$). However, more participants discovered SoftCuts when they declared having to know/use more hotkeys (Figure 4.11b). There was a significant main effect of Quantity of hotkey known/used ($\chi^2 = 22.60, p < .001$) on spontaneous discovery rate, with participants who declared knowing/using 6-10 (25.9%), 21-30 (66.7%), and >30 hotkeys (36.4%), achieving a higher rate (all $p < .05$) than those with only 1-5 hotkeys (4.5%).

Enforced discovery rate  Overall, participants discovered SoftCuts at a higher rate of 57.7% (71 out of 123) when they are constrained to do so. We found a total of 123 participants who started Mission2 without using SoftCuts in Mission1. However, there was no main effect of Familiarity, Saliency, hotkeys’ Usage Frequency, Quantity Known/Used, and Usage Duration on enforced discovery rate (all $p > .05$).

Overall discovery rate  Across both missions, participants discovered SoftCuts at a rate of 62.5% (100 out of 160). There was no main effect of Familiarity, Saliency, hotkeys’ Usage Frequency, Quantity Known/Used, and Usage Duration on overall discovery rate (all $p > .05$).

Therefore, neither familiarity of modifier key or saliency seems to boost discoverability of SoftCuts. However, familiarity of hotkeys, which in our study is correlated with the number of hotkeys they already know on desktop computers does help discoverability.

Subjective Rating

Participants rated the usage of SoftCuts across both Familiarity and Saliency similarly high (all medians $\geq 6$), as shown in Figure 4.14. There was no main effect of
FAMILIARITY or SALIENCY on any of the subjective rating (all $p > .05$). This suggests that SoftCuts is overall perceived positively across the conditions.

4.3.7 Discussion

Spontaneous vs. Enforced discovery rate

It is interesting to observe how the discovery rate tripled when SoftCuts is the only pathway for users to complete the task, moving from a Spontaneous discovery rate at 18.1% to Enforced discovery rate at 57.7%. The overall 62.5% discovery rate across both missions shows that participants could discover SoftCuts without any external help despite the short duration (less than 5 minutes) of interaction spent. However, we also consider ways on how to translate these positive trends to the real world. For instance, two participants (one from KB+CMD/CTRL and another from NOKB+CUSTOM condition) suggested “using a different colour” to increase the saliency of the modifier key. This suggestion may boost SoftCuts’ discoverability rate because the trend from our discovery rates also show that the more salient noKB conditions could perform better than its KB counterparts. We did not choose a contrasting colour for the modifier key because it is important to preserve the aesthetic integrity and consistency of the user interface designs.

Hotkey usage drives discoverability regardless of modifier key label

Our quantitative results reveal that a familiar modifier key does not affect the discoverability rate of our participants, but those who declared knowing/using a higher quantity of hotkey are more likely to discover SoftCuts. It is surprising because we initially assumed that $\dfrac{\text{Ctrl}}{\text{shift}}$ is visually more familiar to users than a custom key. In fact, it is the experience of using hotkeys that increase users inclination to discover by chance. We hypothesise that there is a potential difference between the visual and kinesthetic familiarity of users. This is because an expert who has been accustomed to invoking hotkeys may not need to pay explicit attention to what label was printed on the modifier key, as long as they could remember its position to place their finger, which is more important in this case.

Qualitative benefits of using a familiar modifier key

There were 4 participants who used SoftCuts with the custom modifier key and suggested that its function remained unclear. Particularly, one of them commented, “I only clicked on the $\dfrac{\text{F}}{\text{F}}$ icon out of curiosity as to what it was. I think if I had knowledge of installing it for myself, I would have been familiar and known how to discover it.” Another one of them from the KB+CUSTOM condition also suggested that “there should’ve been a short intro to what the symbol meant”. However, promoting a feature during the onboarding process can do more harm than good and hence we should consider avoiding it whenever possible (Joyce, 2020b). Joyce (2020a) conducted a quantitative usability study that demonstrated how tutorials did not improve task completion.

---

5https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/
time and were even perceived to make the task overly complicated. A solution worth trying to enhance the discoverability of SoftCuts is to use contextual tips, where users could be suggested to explore the more efficient SoftCuts after using the default menus for a long enough duration. Therefore, a familiar modifier key like \texttt{[⌘]} could at least communicate its meaning to users who have prior knowledge of hotkeys, as compared to \texttt{[⌥]} whose function has yet to be known by anyone.

**Overall positive feedback from crowdworkers**

We were surprised to receive detailed feedback from numerous participants when they could easily choose to write a one-word answer for the open-ended final section of this survey. Particularly, 5 participants (1 from \texttt{NO KB+C MD} and 4 from \texttt{NO KB+CUSTOM}) similarly commended the compact keyboard layout used to organise the commands. For instance, one of them said, “I love how every command I need in the Notes app would be in one small little package on my keyboard. So simple and smart”. In addition, 15 participants expressed interests to see SoftCuts integrated into their personal devices when they said, “I would really like to have SoftCuts on my iPhone since I don’t know how to do the things on my notes the way I was able to do it on this demo” or “I would use it right away if it was implemented. Please implement this feature”. Future work should consider launching SoftCuts as a third party keyboard plugin before each company like Apple and Google is ready to integrate SoftCuts into their respective ecosystem.

**4.3.8 Limitations**

Given the multidimensional nature (Posch et al., 2019) of crowdworkers’ motivations, we also acknowledge that our results might demonstrate different results as compared to participants recruited in a lab study. For instance, in terms of the quantity of hotkey known/used, there is a possibility that crowdworkers were not accurate with their answers. Perhaps a more accurate relationship between familiarity and discoverability could be established if there was a way to guarantee a reliable assessment of crowdworkers’ knowledge of hotkey mappings, which we are not aware of. Future studies should consider evaluating the discoverability of SoftCuts with participants in the lab to generalise our results further.

**4.4 Conclusion**

We learn from this chapter how visual elements of SoftCuts could be designed to maximise its discoverability. Our Study 1 results revealed that while D1 (only letters) was the least preferred, D3 (command names and icons) was most preferred, and D2 (command names and letters) could be the next best alternative after D3. Our Study 2 results revealed that the familiarity and saliency of the modifier key do not affect the discoverability for SoftCuts. However, if a user is familiar with using hotkeys on a desktop, our results confirm that they are more likely to discover SoftCuts. Therefore, we recommend that designers consider using a keyboardless option when the active task requires as much screen space as possible and only use the full keyboard mode.
sparingly. The insights from this chapter help address one of the concerns raised by our thesis statement (see Chapter 1.2.2), that is, to consider the needs of novice users who may struggle to discover a new feature like SoftCuts.
Chapter 5

Evaluating Input Methods for SoftCuts

5.1 Motivation

Beyond maximising the usability for novice users, it is equally important to ensure that user performance and adoption of SoftCuts is not being compromised regardless of expertise. At the heart of it, there are three different input methods that are key candidates we consider while designing SoftCuts.

1. **User-Maintained interaction (UM)**: Holding the modifier key with one finger and activating the hotkey with a different finger. This mechanism is similar to hotkeys on desktop computers and is also used on Samsung Galaxy tablets (Samsung, n.d.).

2. **Two sequential taps (Once)**: Tapping the modifier key (without holding) changes the keyboard layout to display commands that can then be selected. Subsequent tapping on any key will change the layout back to QWERTY automatically. This mechanism is similar to how the `Shift` key behaves to capitalise text with soft keyboards and how hotkeys are implemented in recent Microsoft Surface tablets (PandaSage1221, 2013).

3. **Sliding gestures (Swipe)**: Touching the modifier changes the keyboard layout to display the commands, which can be selected by sliding the finger over the key and releasing it. This mechanism is similar to the currently discontinued Swype keyboard’s implementation (Nuance, 2017).

We illustrate how these three input methods are compatible with each other in the state diagram (see Fig. 5.1), leaving users the freedom to choose their preferred method depending on the context. In this chapter, we focus on investigating their performance and usage through two studies. First, we compare the performance of (Once, Swipe and User Maintained (UM)) for SoftCuts across different devices, orientations and number of hands used to interact regardless of the visual presentation. While a previous study (El Batran and Dunlop, 2014) theoretically estimated that a swipe is approximately 10ms faster than a single pointing gesture, it is unclear how that result would translate to our unique context of keyboard layout and two sequential taps instead of a single one. Second, we evaluate the usage of the same three input methods across varied mobility conditions like sitting, standing and walking. Which method performs the best?
Do people switch methods as they wish, or is there a particular one they would prefer? How will the results change in different mobility conditions? The answers will be revealed next.

5.2 Study 1: Performance of Input Methods

5.2.1 Participants and Apparatus

Twelve participants (all right-handed, 5 female), aged 18 to 30 (M=25.1, SD=3.29) were recruited from the university community. They received a S$10.00 reimbursement for their 45 minutes participation. One participant used a tablet every day while three used it at least once per month. Only one participant was familiar with swipe-based keyboards. The experimental software was written in Java using Android Studio as an IDE and ran on version 28 of Android SDK. We used a Samsung A10 phone (6.2", 168 grams) running Android 9 for the Phone condition and a Samsung Galaxy Tab 4 (10.5" screen, 483 grams) running Android 9 for the Tablet condition.

5.2.2 Procedure

Participants began the experiment with a questionnaire that asked about their demographic information, familiarity with tablet and use of swipe-based keyboards. They were then briefed on the study and were presented the three input methods. They would then be asked to perform repeated series of command execution as rapidly and accurately as possible, with a given input method and a given configuration. A configuration corresponds to the combination of an assigned device (phone or tablet), an assigned orientation (portrait or landscape) and an assigned handedness (one or two hands). The configurations compared in this study are detailed in Fig. 5.1. We excluded
TABLE 5.1: Experimental Conditions for Study 1. Greyed-out conditions were excluded.

<table>
<thead>
<tr>
<th>Hands</th>
<th>Device</th>
<th>Orientation</th>
<th>User Maintained</th>
<th>Once</th>
<th>Swipe</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Phone</td>
<td>Portrait</td>
<td>2 Hands Required</td>
<td></td>
<td>1-handed part</td>
</tr>
<tr>
<td></td>
<td>Tablet</td>
<td>Landscape</td>
<td>Physical Constraints</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Phone</td>
<td>Portrait</td>
<td></td>
<td></td>
<td>2-handed part</td>
</tr>
<tr>
<td></td>
<td>Tablet</td>
<td>Landscape</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.2: Example of a trial on smartphone. (a) The stimulus is highlighted in blue at the beginning of the trial to minimise visual search (b) the user pressed on one of the modifier keys, (c) upon correct selection, the key would turn green (red if incorrect).

UM in 1-handed conditions because it requires two hands. Gutwin, Cockburn, J. Scarr, et al. (2014)’s FastTap is one example of UM which requires one hand to support the device and another hand to perform the chording. During pre-tests, we also found out that participants were not able to reach every key in the 1-handed Phone-Landscape conditions. Hence, they were discarded as well.

5.2.3 Tasks

As we were interested in the performance of command execution and wanted to eliminate other factors such as visual search, each trial began by specifying the target command by highlighting its key in blue (Fig. 5.2a). This is a common practice in command selection experiments (Bailly, Lecolinet, and Nigay, 2016). Participants then had to select the target command using the assigned input method. In that respect, participants could select the commands using either of the two modifier keys, respectively located on the bottom left and right corners of the keyboard (Fig. 5.2). After performing the selection, feedback would appear on the screen for 500 milliseconds, with the target being highlighted in either green (if correct) or red (if incorrect) before proceeding to the next trial (Fig. 5.2c). A trial was considered successful if the participant was able to select the correct target. Trial completion time was measured from target appearance to target selection.
5.2.4 Stimuli

There were 6 possible targets (Q, Z, T, V, P and M) as these keys are spread around the keyboard. For each condition (Device × Orientation × Handedness combination), participants repeated 1 training block of 6 trials (once each target, in random order), followed by 3 test blocks of 12 trials (twice each target, in random order). After each condition, participants provided subjective feedback on a 1-5 Likert Scale (1: strongly disagree, 5: strongly agree) on their overall opinion for the method (“I liked using this input method”), ease of use, comfort, perceived speed and perceived accuracy. Participants then proceeded to the following condition for that part. For the full details of the questions we asked, please refer to Appendix C.

5.2.5 Design

Our initial design space for this study includes:

**INPUT METHOD**
- UM, ONCE, SWIPE

**DEVICE**
- PHONE, TABLET

**HANDEDNESS**
- ONE-HANDED, TWO-HANDED

**ORIENTATION**
- LANDSCAPE, PORTRAIT

Complete factorial exploration of this design space would lead to a total of $3 \times 2 \times 2 \times 2 = 24$ combinations. However, we decided to discard 10 combinations that we considered impractical or impossible, focusing our study on 14 combinations that were considered comfortable and/or feasible (see Table 5.1). The study conditions were split into two parts (one- and two-handed). The order of appearance of both parts was fully counterbalanced across participants.

**One-Handed Conditions**

For **ONE-HANDED** conditions, participants were invited to hold a PHONE in PORTRAIT orientation only with their preferred hand. TABLET device with LANDSCAPE orientations were discarded either for physical constraints or because two hands were required for comfortable interaction. Similarly, UM was excluded as it requires two simultaneous contact points, which cannot be achieved in these configurations. This results in a within-subject design with **INPUT METHOD** as a two-level { ONCE, SWIPE } independent variable, fully counterbalanced across participants. In total, a participant would perform $2 \times [(1 \text{ training block } \times 6 \text{ stimuli}) + (3 \text{ test blocks } \times 6 \text{ stimuli } \times 2 \text{ repetitions})] = 84$ trials.

**Two-Handed Conditions**

For two-handed conditions, we have a $3 \times 2 \times 2$ within-subject design with three independent variables: **INPUT METHOD** { UM, ONCE, SWIPE }, **DEVICE** { PHONE, TABLET }, and **ORIENTATION** { LANDSCAPE, PORTRAIT }. Participants were instructed to hold the device with their non-preferred hand and were instructed that they could use both hands to perform the selections. We did not restrict the specific hand posture, as long
as the device was held at least with the non-preferred hand. Input method was counterbalanced using Latin Square, and Device and Orientation were collapsed into four conditions counterbalanced using Latin Square (Phone-Landscape, Phone-Portrait, Tablet-Landscape, Tablet-Portrait). A participant would perform 3 input methods × 2 devices × 2 orientations × [(1 training block × 6 stimuli) + (3 test blocks × 6 stimuli × 2 repetitions)] = 504 trials in the two-handed part.

**Dependent Variables**

We measured command selection accuracy and time. A trial was considered successful if the participant could select the correct target on the first attempt. Selection time was measured as the time between the stimulus presentation until the user finished the selection (release of hotkey for UM and Once, end of the slide gesture for Swipe). The modifier key recorded was the last one used before finishing the trial (left or right). For each combination of input method × device × orientation, we also measured subjective feedback on a 1-5 Likert Scale (1: strongly disagree, 5: strongly agree) on their overall opinion for each method (“I liked using this method”), ease of use, comfort, perceived speed and perceived accuracy.

Each participant spent approximately 45 minutes completing the study and was allowed to take breaks between each block. In total, we recorded 12 participants × 84 trials in one-handed + 504 trials in two-handed = 7056 trials overall.

**5.2.6 Results**

We conducted two separate statistical analyses for one-handed and two-handed parts. First, we marked each trial as an outlier if its trial time was above or below 3 standard deviations from the mean time. A total of 46 trials (0.65%) were removed for subsequent analysis. Then, we checked the distribution of our time measurement and found one-handed part normally distributed, while the two-handed part did not, which we then applied log transformations. For the one-handed part, we used paired t-test for accuracy and time (as we only had two input methods). For the two-handed part, we used three-way ANOVA with repeated measures for time, with pairwise t-tests with Bonferroni corrections for post-hoc comparisons. With respect to accuracy, we used Aligned Rank Transform (Wobbrock, Findlater, et al., 2011). We applied Greenhouse-Geisser sphericity correction when needed, which corrects both p-values and the reported degrees of freedom. For subjective measurement, we used the Wilcoxon signed-rank test for analysis. We do not compare performance between one-handed and two-handed parts, as the design and number of trials differ. The 95% confidence intervals shown in the Figures are computed using the z-scores with the following formula: $1.96 \times \frac{SD}{\sqrt{n}}$.

**One-handed Part**

Participants performed the task rather quickly ($M=1.11s$) and accurately ($M=95.8\%$), as seen in Fig. 5.3. While average command selection time was lower with ONCE ($M=1.08s$) compared to SWIPE ($M=1.15s$), the difference was not significant ($p = .264$). However, we found a significant effect of input method on accuracy ($t(11) = 3.86, p < .01$), as
our participants reached higher accuracy with ONCE ($M = 98.4\%$) than with SWIPE ($M = 93.3\%$). In terms of subjective preferences, we did not observe any significant effect of INPUT METHOD on any question. The median rating is 4 for ONCE and 3 for SWIPE. This suggests that both input methods are overall perceived positively for one-handed configuration.

Two-handed Part

**Time** We found a significant main effect of INPUT METHOD on time ($F_{2,22} = 23.9$, $p < .0001$, $\eta^2_G = .13$). Participants were faster with ONCE ($M = 0.89\ s$) than with SWIPE ($M = 1.07\ s$, $p < .0001$) and UM ($M = 1.04\ s$, $p < .0001$). We also found a significant main effect of ORIENTATION ($F_{1,11} = 8.00$, $p = .016$, $\eta^2_G = .01$), with a slightly lower average time in PORTRAIT conditions ($M = 0.97\ s$ vs. $M = 1.03\ s$ for LANDSCAPE). There was a significant main effect of DEVICE ($F_{1,11} = 8.09$, $p = .016$, $\eta^2_G = .01$), with a slightly
lower average time in PHONE conditions (\(M = 0.97\) s vs. \(M = 1.03\) s for TABLET).

We found a significant DEVICE \(\times\) INPUT METHOD interaction effect on time (\(F_{2,22} = 5.35, p = .013, \eta^2 = .03\)), with UM performing better on PHONE than on TABLET, and SWIPE following the opposite pattern (Fig. 5.4). There was also a significant DEVICE \(\times\) ORIENTATION interaction effect on time (\(F_{1,11} = 26.4, p < .001, \eta^2 = .04\)), with PORTRAIT being faster on PHONE, while LANDSCAPE being slightly faster on TABLET (Fig. 5.5). We did not find any other interaction effect (all \(p > .05\)).

**Accuracy** INPUT METHOD had a significant effect on accuracy (\(F(2,121) = 39.1, p < .0001\)). Our participants were significantly less accurate when using SWIPE (\(M = 94.5\%\)) compared to ONCE (\(M = 99.4\%, p < .0001\)) and UM (\(M = 99.5\%, p < .0001\)). We
did not find any significant main effect of ORIENTATION \( (p = .85) \) on accuracy.

We found a significant DEVICE \( \times \) INPUT METHOD interaction effect on accuracy \( (F(2, 121) = 10.9, \ p < .0001) \), with SWIPE being much less accurate on PHONE than on TABLET (Fig. 5.6). The average accuracy was significantly higher \( (F(1, 121) = 22.1, \ p < .0001) \) on TABLET \( (M = 98.7\%) \) than on the PHONE \( (M = 96.9\%) \).

**Modifier Keys** Our participants used the left modifier key more often. For ONCE, they used it a total of 1103 times (vs. 625 for Right). Usage was similar for SWIPE (1108 Left vs. 620 right). The trend is even clearer for UM (1265 trials vs. 463 right). This is likely due to the constrained posture on the left hand, where they could easily access the Left key with their thumbs.

**Subjective Preferences** After each condition of our study, we gathered the subjective preferences of our users (Fig. 5.7). For PHONE, participants perceived the three INPUT METHODS as similarly positive across all the questions asked. For TABLET, participants rated ONCE (median=4.0) more favourably than UM (median=3.0) in terms of like, speed, and comfort. Participants also rated ONCE (median=5.0) more favourably than SWIPE (median=3.0) in terms of like and speed.

The results from this study are discussed in Section 5.3.7, along with the results from the next study.

### 5.3 Study 2: Usage of Input Methods

In this study, our goal is to observe which input method is to be adopted when the user is not constrained and across different activities like sitting, standing and walking. Since the 3 input methods are compatible with one another, the users have the freedom
to use the one they want depending on their preferences for each specific scenario. We are interested to see how the results from Study 1 would translate when evaluated in a more realistic environment.

5.3.1 Participants and Apparatus

Twelve participants (all right-handed, 5 female), aged 21 to 28 (\( M = 24.2, SD = 2.72 \)) were recruited from the university community. They received a S$10.00 reimbursement for their 45 minutes participation. None of these participants took part in the previous study. Only one participant had never used a tablet prior, and we used the same phone and tablet as in Study 1. In addition, some conditions required the use of a treadmill (see Figure 5.8) to simulate a consistent walking speed of 2.5 km/h as in previous works (Roumen, Perrault, and Zhao, 2015; Je et al., 2018).

5.3.2 Procedure

Participants began with completing a questionnaire about their demographic information. They were then briefed on the study and started the practice phase. During this practice phase, participants were asked to perform 16 command selections with each input method while seated down. The command selection was completed on both phone and tablet, with two hands and in portrait mode. The order of presentation of the three input methods was fully counterbalanced across participants to avoid any bias towards one specific input method. The goal of the practice phase was to give participants a chance to familiarise themselves with each input method before starting the study.
After practising, participants were briefed on the actual study, in which they had to complete three blocks per individual condition (see Design section). Each block has 16 trials (twice each target, in random order). Participants were instructed to perform the command selection with the input method they preferred and would rather use in real-life cases. They were also made aware that they were free to change their choice of input method during the study as they wished. After finishing all conditions, participants were asked to explain their choices of input method to the experimenter in a short semi-structured interview. For the full details of the questions we asked, please refer to Appendix D.

5.3.3 Soft Keyboard Layout

Similar to Study 1, the soft keyboard used in this study has command keys located at its bottom-left and right corners (Fig. 5.9). When either of the command keys was hit, the soft keyboard will be displayed instantly. Keys with associated commands would be rendered with both their icons and names, while keys without associated commands would be greyed out. This choice of graphical representation is further supported by our study in Chapter 4. We also chose countries from different continents that have consistent flag shapes, similar to previous studies (Malacria, Bailly, et al., 2013; Goguey, Malacria, Cockburn, et al., 2019b). The layout for the 16 commands is based on the 16 most common hotkeys on MS Word, which we then mirrored on the vertical axis to preserve spatial distribution and have a realistic distribution of commands based on an existing layout.

5.3.4 Task and Stimulus

Each trial began with the name of a target command displayed on top of the screen. Then, participants would press the modifier key and select the corresponding command key using any of the three available input methods. Upon selection, either a positive (in green) or negative (in red) feedback would highlight the selected key for 500 milliseconds before proceeding to the next trial. For each participant, 8 of the 16 commands were randomly used as stimuli, similar to FastTap (Gutwin, Cockburn, J. Scarr, et al., 2014). To ensure similar spatial distribution of selected commands, 2 out of
the 8 were from the top row and 3 from each middle and bottom row. We also avoided mnemonic associations between keys and countries, e.g. "A" for Australia.

5.3.5 Design

Our initial design space for this study includes: ACTIVITY \{SITTING, STANDING, WALKING\}, DEVICE \{PHONE, TABLET\}, HANDEDNESS \{1-HANDED (1H), 2-HANDED (2H)\}, ORIENTATION \{LANDSCAPE, PORTRAIT\}. Complete factorial exploration of this design space would lead to a total of $3 \times 2 \times 2 = 24$ combinations. Participants were asked to hold both devices, similar to Study 1. However, to keep the study within a manageable time frame, we decided to prioritise some factors depending on the device used. We thus discarded some factors for either PHONE or TABLET conditions.

For the PHONE conditions, we deemed HANDEDNESS as more important than ORIENTATION, as one- and two-handed conditions may yield different results. On the other hand, we did not envision many scenarios with a physical activity where landscape orientation would be extensively used and thus decided to set the ORIENTATION to PORTRAIT mode and included HANDEDNESS in our design. For the TABLET conditions, we assumed that most interactions with a tablet would be two-handed while in motion or simply because supporting the weight with only one hand can result in fatigue. As such, we set the HANDEDNESS to 2H and included ORIENTATION in our design. As such, the PHONE and TABLET conditions are not directly comparable as the impact of HANDEDNESS and ORIENTATION is different. We will thus conduct separate analysis for each device.

In this study, we measure the usage of each INPUT METHOD for our different conditions. For each possible INPUT METHOD, we count the number of trials where command selection was performed using that specific method. Simultaneously, we also measured the corresponding time taken and accuracy (i.e. whether the correct command was selected) for each trial. The order of presentation of DEVICE, HANDEDNESS (Phone) and ORIENTATION (Tablet) was fully counterbalanced across participants, while ACTIVITY was counterbalanced using Latin Square.

Each participant took approximately one hour to complete the study. Participants were allowed to take breaks between conditions. In summary, we recorded 12 participants $\times$ 3 activities $\times$ 2 devices $\times$ 2 handedness OR orientation $\times$ 3 blocks $\times$ 8 commands $\times$ 2 repetitions = 6912 trials in total.

5.3.6 Results

The goal of this study is to find out which input method participants prefer to use, thus we report the usage frequency from our study. The results are overall clear, with one technique being chosen most of the time. We were also interested in finding out whether ACTIVITY, HANDEDNESS and ORIENTATION impact the usage frequency. To do so, we performed a multinomial logistic regression with R by giving numerical values to our ACTIVITY levels (1: sitting, 2: standing, 3: walking) and ORIENTATION (1: portrait, 2: landscape). We separated the analysis into three parts: one for Phone 1-Handed
Chapter 5. Evaluating Input Methods for SoftCuts

Figure 5.10: Distribution of each Input Method (UM, Once, Swipe) across 4 conditions between device, handedness, orientation while sitting, standing and walking. (a) is derived from actual usage during the study while (b) is the declared preference at the end of each condition.

(as UM is not usable here), one for Phone 2-Handed, and one for Tablet.

From our results, we then compared the time and accuracy performance. However, given the small number of samples for Swipe and UM, we only performed statistical analysis for time and accuracy for the Once method, using ANOVA.

Usage Frequency

Overall, our users mainly used **ONCE**, with this method used for 86.2% of selections. Its usage proportion was at least 76.6% (Phone 2-Handed Sitting) and at most 91.3% (Tablet Landscape Walking). The usage frequency of each technique across the different conditions is shown in Fig. 5.10.

**Phone 1-Handed** Participants performed most trials using **ONCE** (84.3% of selections) and used **SWIPE** for the remaining trials (15.7%). Our analysis shows that **ACTIVITY** impacted the usage frequency of both techniques (both \( p < .001 \)): the usage of **SWIPE** and **ONCE** varies with **ACTIVITY**, from 18.2%-81.8% sittin to 16.3%-83.7% standing and 12.5%-87.5% walking.

**Phone 2-Handed** **ONCE** was again the most frequently used (82.5% of selections), with a significant effect of **ACTIVITY** \( (p < .001) \) with usage ranging from 76.6% while **SITTING**, to 83.5% while **WALKING** and 87.2% while **STANDING**. **SWIPE** was the second most used at a steady 11.7% of usage, ranging from 11.1% (sitting) to 12.2% (walking) with no effect of **ACTIVITY** \( (p > .05) \). Finally, **UM** was used for 5.9% of selections, and was strongly impacted by **ACTIVITY** \( (p < .001) \) with little to no usage while **STANDING** (1%), rather limited while **WALKING** (5.9% usage) and to a level comparable to **SWIPE** while **SITTING** (11.3%).

**Tablet** **ONCE** was overall used for 89% of the selections, and its usage was not impacted by **ORIENTATION** or **ACTIVITY** (both \( p > .05 \)). **UM** was the second most chosen
input method, used for 10% of the selections consistently across conditions with no visible effect of ORIENTATION or ACTIVITY (both $p > .05$). Swipe was only used for 0.98% of the trials with no visible effect of ACTIVITY on its usage. However, we did notice an effect of ORIENTATION on SWIPE usage ($p < .01$): our participants nearly did not use SWIPE in PORTRAIT mode (0.3% of selections) and used it a bit more in LANDSCAPE (1.6% of selections).

**Individual Participants Usage** Among our participants, we did notice that one specific participant (P8) used SWIPE for 98.9% of their trials on the PHONE and UM for 99.3% of their trials on TABLET. The participant reported that they did not like the ONCE method overall and that they were used to SWIPE for typing on soft keyboards on their phone. This suggests that exposure to input methods may have an impact on usage frequency. Our other participants showed a consistent pattern of relying mostly on ONCE for 90% selections or more and usually trying out the other techniques across conditions.

**Time and Accuracy**

We also analysed the objective performance by filtering trials that were completed using ONCE. We could not analyse for the other two techniques because of the lack of data available for some conditions, as these techniques were seldom used (e.g. SWIPE on TABLET or UM on PHONE). We performed one analysis for each device, because as mentioned in the Design section above, we included HANDEDNESS for phone’s design space and ORIENTATION for tablet’s. A total of 119 trials (2%) were marked as outliers and removed for subsequent analysis.

**Phone** In terms of time, we found significant effect of HANDEDNESS ($F_{1,10} = 11.44$, $p < .01$, $\eta^2_p = .99$) on phone’s performance. It took participants on average 1.51s to complete a trial on the phone using 1 hand, and 1.35s using 2 hands. We did not find any effect of ACTIVITY or interaction (both $p > .05$). For accuracy, we did not observe any effect of HANDEDNESS, ACTIVITY or interactions (all $p > .05$), with an average accuracy of 98.5%.

**Tablet** For the tablet conditions, there was no significant effect of ACTIVITY, ORIENTATION and no interaction on time (all $p > .05$). On average, a trial would last for 1.38s to complete, which is very similar to that of phone’s 2-handed performance. In terms of accuracy, there was no significant effect of ACTIVITY, ORIENTATION and no interaction (all $p > .05$), with participants reaching 98.7% accuracy on average.

The time and accuracy results seem to be in line with that of Study 1 (Fig. 5.4), where the performance of Once is comparable between Phone-2H (0.86s) and Tablet-2H (0.91s). The task in Study 1 was a simple pointing task, while the task in Study 2 required the user to perform a visual search to identify the correct command. The effect of HANDEDNESS on time for the phone condition seems consistent with the average performance observed in Study 1, where users were overall faster in 2-handed mode (0.86s) vs. 1-handed mode (1.08s). An interesting result from Study 2 is that the
performance of Once does not seem to be affected by activity, making Once a robust input method for real-life scenarios.

5.3.7 Discussion

We now discuss our results, with the strengths of each input method and what these results mean for SoftCuts. For each input method, we also discuss how the technique could be adapted to perform multiple hotkey selections sequentially.

Is Once the best input method for SoftCuts?

In our first study, Once achieved the overall highest performance in terms of time, accuracy and preference. Once significantly stands out, yielding the lowest command selection time both on tablet and phone, regardless of device orientation and how they were held. Participants also achieved near-perfect accuracy of 98-99% in both studies. From Study 2, we found out that Once was chosen as an input method near 4 out of 5 trials, making it the most used input method, no matter the device, handedness or tablet orientation. More importantly, the performance of this method did not seem to be affected by activity, making Once the best input method.

Several participants (P1,5,6,7) shared the similar sentiment that Once is "versatile" because of its two discrete steps, unlike in UM having to communicate between 2 finger actions, which requires more "cognitive effort to use" (P12). That being said, we must stress that both of our studies required users to perform a single command selection per trial, which appears to be the best-case scenario for this technique. Indeed, Once relies on a pseudo-mode (unlike UM which relies on a user-maintained mode). Therefore, it may be less efficient and more tedious to perform several command selections at once. This is given that the keyboard would switch back to text entry mode after each command selection, hence requiring users to reactivate command mode for each subsequent command selection. One way around that would be to allow users to double-tap on the modifier key, which would lock the mode to command selection, the same way that one can double-tap the Shift key on a soft keyboard to maintain the upper case mode.

We had to make experimental decisions and trade-offs to keep the length of our studies under acceptable duration. Future work should investigate the impact of the number of commands selected per trial on input method performance and usage.

Swipe as a suitable Phone input method

Swipe was a compelling case to investigate. In Study 1, we showed that its performance was objectively worse than other input methods. The gap was more significant on the phone, where we observed a lower accuracy overall (around 92% on average in both 1- and 2-handed), as well as a larger selection time. The performance of Swipe on the tablet was closer to the other techniques, and thus, we expected to see better adoption of Swipe on tablets. However, it appears that users did not want to use Swipe on the tablet, as it was used for around 1% of the selections and even less in landscape mode.
P4,5,7,8,11,12 found dragging across the larger screen area of the tablet was "challenging" and "uncomfortable". Participants reported that swiping was too "tedious on the tablet", usually because of "the long distance that [one] needs to swipe" (P4). P8 also reported that swiping allowed them to "change their decision before releasing the gesture at the right key", suggesting that Swipe allows user to partially recover from erroneous selections, as long as they did not release their finger.

Despite these issues, Swipe was the second most used input method for phones, used for 15.7% of the selections in Phone 1-handed, and 11.7% Phone 2-handed. One participant even exclusively used Swipe, citing their significant usage of swipe based text entry methods and their strong dislike of Once. All users did perform part of their selections using Swipe on the phone nonetheless. This suggests that Swipe could still be the primary input method on the phone for specific categories of users, especially in 1-handed scenarios, despite its objectively worse performance. It is also worth noting that participants tended to rely on Swipe more on sitting and standing positions, while the selection was deemed harder while walking. Swipe, by default, does not support multiple selections in a row. It could support that using, for example, a dwell mechanism: users willing to select a specific command would stop on it for a short duration, then proceed to the following command. However, this would likely put the technique at a substantial disadvantage compared to Once and even more to UM.

**User Maintained as a Tablet input method**

UM is the only of our input methods that require two hands to operate. However, its performance in Study 1 is close to Once in terms of speed and accuracy, with a small difference of 100 ms for Phone conditions. On the other hand, its subjective preference was overall the lowest, which led us to believe that UM may not be used at all.

We were once again surprised to find out that UM was used in Study 2. In Tablet mode, specifically, it was chosen around 10% of the time. P5 commented about an ergonomic advantage that they felt with UM while supporting the relatively heavy weight of the tablet. They said, "as the thumb holds onto the modifier key at the bottom, a grip action is resembled, hence making it easier and more steady to use the other hand’s finger to trigger the (target) key." Another user also performed most of their selection with this method because of their dislike of Once.

Another strong advantage of UM over the other techniques is that it readily supports multiple selections, as illustrated in Fig. 5.1. It is important to note that both our studies featured only a single command selection per trial, putting UM in a worst-case scenario. Thus, we could expect UM to be more widely used for productivity tasks, where users select multiple commands at once.

**SoftCuts in real world**

Both studies show that users can quickly select commands, with an average time of around 0.89s in Study 1 (pointing only) vs. 1.41s in Study 2 for Once. Accuracy was also extremely high, and users enjoyed using the system, as seen in the high score of the
subjective preferences. This makes SoftCuts a good candidate for command selection technique on both tablet and phones. Our vision for SoftCuts is illustrated in Fig. 9.1.

A minimal way to implement SoftCuts on mobile devices is to provide Once as an input method. However, as highlighted above, some users are likely willing to use Swipe on their phone, and UM already supports multiple command selection. Based on our state machine presented in Fig. 5.1, the three input methods are compatible, allowing designers to implement all three of them without interference.

5.4 Conclusion

Our results suggest that while the input method Once (based on two successive taps on soft keys) overall performed best, some users also like using Swipe for phone and UM for tablet interaction. Altogether, these results confirm that SoftCuts could be generalised as an efficient command selection mechanism for touch-based devices. Besides determining an efficient input method for SoftCuts, the learnability of command mapping is another factor that could affect the performance of selections. We have yet to understand the impact of SoftCuts in terms of learnability and this is what the next chapter 6 will be focusing on.
Chapter 6

Leveraging Prior Knowledge and Sustaining Retention

6.1 Motivation

One advantage of SoftCuts over existing command selection techniques is that it is similar to desktop hotkeys. As such, it is possible to use on touch-based devices the same command/key mapping as that on desktop computers, with which users may be familiar and able to capitalise. This would provide a substantial advantage to users familiar with these existing mappings, their selection performance, and help long-term retention. However, not every mobile application has a corresponding desktop application from which it could get its mapping. Thus, SoftCuts will still need users to learn new command/key mappings for these applications. Thus, we are interested in finding out how easy it is for users to learn SoftCuts, and how prior knowledge helps them in this learning process.

6.2 Soft Keyboard Layout

The soft keyboard used in this study had the modifier keys located at its bottom-left and right corners (Fig. 6.1 and 6.2). We chose an abstract circle icon instead of the ` ctrl ` used in our previous Study 2 of Chapter 4 to avoid any potential confounding effect. We ensured that participants are aware that these circle icons represent the modifier key. When either of the modifier keys was hit, the soft keyboard would be displayed instantly. Keys with associated commands would be rendered with both its icon and name. Keys without associated commands would be greyed out. This graphical representation is different from commercial solutions, highlighting the key (for Samsung) or revealing the command name at the top of the key (for Microsoft Surface). It was chosen based on the results from our previous Study 1 of Chapter 4 and that icons act as an efficient visual cue for guiding visual search Bailly, Lecolinet, and Nigay, 2016 and conveying command meaning Giannisakis et al., 2017.

6.3 Task

This study relied on a simple command selection task. At the beginning of each trial, the target command name would be displayed on top of the screen (Fig. 6.1a). Participants thus had to select the command by hitting the modifier key to reveal the soft
Figure 6.1: Example of a trial in the experiment. (a) The participant was first presented with the stimulus "Australia", (b) they clicked on one of the left/right modifier keys at the bottom of the screen, then (c) they accurately selected "Australia" and received feedback. The abstract command set is shown on the (b) and (c) panels.

Figure 6.2: Layout of our realistic command set, taken from the Windows version of Microsoft Word.

keyboard (Fig. 6.1b) and then pressing the target command key. Upon selection, participants would receive both audio and visual feedback for 500ms to indicate success or failure (Fig. 6.1c). Participants were instructed to perform the task as fast and accurately as possible. For each trial, time was measured as the time between stimulus appearance to when the correct target command was selected.

6.4 Stimuli and Command Mappings

We tested two different command mappings (ABSTRACT and REALISTIC) of 16 commands each. The abstract set is made up of 16 countries (Fig. 6.1b,c), while the realistic set is made up of 16 existing hotkeys from Microsoft Word for Windows (Fig. 6.2). The spatial layout of the abstract command layout was a vertical symmetry of the realistic command one. This ensures a similar distribution in rows and columns and a comparable average distance between the modifier keys and the command keys. We also avoided mnemonic associations between keys and countries, e.g. [A] for Australia. 8 commands from each set were chosen as targets: Australia, Brazil, Canada, Denmark, Germany, Iran, South Africa, and Vietnam for abstract set and Add Link, Cut, Find, Paste, Print, Save, Underline, and Undo for the realistic set. We decided to avoid having the same key used as a target in both conditions (e.g. [X] mapped to United States and Cut) to prevent users from potentially creating memorisation strategies by associating realistic commands with abstract commands.


Chapter 6. Leveraging Prior Knowledge and Sustaining Retention

6.5 Procedure

Participants began the experiment by signing a consent form and completing a questionnaire about their demographic information, handedness, tablet usage, and Microsoft Word usage. They were then evaluated on their knowledge of the 16 commands from the realistic set. For each mapping condition, learning is facilitated when participants performed 1 training block (B0) followed by 8 test blocks (B1-B8). Each block consists of 16 trials, where each of the 8 chosen target commands was repeated twice in random order. We ensured that participants did not accumulate fatigue by recommending ample breaks between blocks. Participants would also need to provide subjective ratings on each mapping after the completion of each condition. Finally, we measured retention rates on the 8 abstract and realistic commands at multiple phases: at the end of the study trials and after 1, 3 and 7 days. Each retention phase is facilitated through a form and requires less than 2 minutes. In the form, we displayed an ordinary keyboard layout and the 16 target commands (with icons) used during trials (8 for each mapping), and we asked participants for the keys mapped to each command. For the full details of the questions we asked, please refer to Appendix E.

6.6 Participants and Apparatus

We recruited 12 participants (8 female, 4 male, all right-handed), aged 19 to 29 ($M=24.9$, $SD=2.9$) from the university community. All 12 participants were not involved in previous experiments. All of them had used Microsoft Word before, while 4 of them had never used a tablet. The experimental software was written in Java using Android Studio as an IDE and ran on version 28 of Android SDK. We used a Samsung Galaxy Tab 4 (10.5” screen, 483 grams) running Android 9 for the experiment, and each participant

We asked each participant if they knew the hotkey mapped to the 16 commands from our REALISTIC set. None of the 12 participants knew the shortcut mapped to Add Link command $K$, 9 knew both Underline $U$ and Undo $Z$, and all of them knew the rest of the commands. This means that out of the 8 target commands, all our participants had to learn only one of them (i.e. Add Link). 3 participants had difficulties assigning the key $U$ between Underline and Undo commands since both share the same starting letter. Participants could rely on prior knowledge and minimise the slower visual search for the remaining 5 targets (Cut, Find, Paste, Print, and Save). On the other hand, participants had no prior knowledge of all 8 target commands for the ABSTRACT set and had to learn them through the trials.

6.7 Design

We used a within-subject design with MAPPING as a primary independent variable with 2 levels { ABSTRACT, REALISTIC }, and BLOCK and PHASE as secondary independent variables. BLOCK has 9 levels { B0-B8 } while PHASE has 4 levels { 0 DAY (0D),
1 DAY (1D), 3 DAYS (3D), 7 DAYS (7D)}. We fully counterbalanced the order of mapping being presented to the participants. In terms of dependent variables, we measured the time and accuracy of each trial and retention rate of each mapping. Note that accuracy measures the learning stage, while retention rate measures the recall stage. A trial was considered successful if a participant was able to select the correct command. We measured accuracy as the ratio between successful trials and the total number of trials for each block. We also measured subjective feedback using a 1-5 Likert Scale (1: strongly disagree, 5: strongly agree) on their overall opinion (“I liked using this mapping”) between abstract and realistic command mapping, in terms of ease of use, speed, accuracy and how this mapping leveraged their prior knowledge of hotkeys. The participants completed both learning and recall stages in approximately 30 minutes over one week. There were no dropouts throughout the multiple sessions. In summary, we recorded 12 participants \( \times 2 \) mappings \( \times (1 \) training block + 8 test blocks \) \( \times 8 \) commands \( \times 2 \) repetitions = 3456 trials in total.

6.8 Results

We first marked each trial as an outlier if its trial time was above or below 3 standard deviations away from the mean time. A total of 63 trials (1.8%) were removed for subsequent time analysis. 32 of which are from abstract mapping and 31 of which are from realistic mapping. Since our time measurements significantly deviated from normality, we applied log transformations. We also transformed accuracy data using Aligned Rank Transform Wobbrock, Findlater, et al., 2011. We used a MAPPING \( \times \) BLOCK ANOVA, followed by pairwise t-tests with Bonferroni corrections for post hoc comparisons. When the assumption of sphericity was violated, we corrected both p-values and degrees of freedom using Greenhouse-Geisser (\( \epsilon < 0.75 \)). For every dependent variable, except subjective rating, trials were aggregated by participant and factors being analysed. For subjective rating, we used the Wilcoxon signed-rank test for analysis.

Learning Effect

There is a significant main effect of block \( (F_{8,88} = 23.40, p < .0001, \eta^2_G = .19) \) on time, but no interaction effects involving block. Pairwise comparisons found that training block 0 took significantly more time to complete than all test blocks 1-8 (all \( p < .001 \)). Hence, we removed block 0 for the main analysis but included a brief analysis of time for training data only below. In the subsequent analysis, we only used the 8 test blocks (1-8) since they are more representative of practised performance.

Time

For test blocks, we found that participants select realistic commands faster than abstract commands. There were significant main effects of mapping \( (F_{1,11} = 7.87, p = .017, \eta^2_G = .43) \) and block \( (F_{7,77} = 2.81, p = .012, \eta^2_G = .03) \) on time but no interaction effect between mapping and block \( (p > .05) \). However, pairwise comparisons did not show any significant difference between the test blocks (all \( p > .05 \)). The significant difference between abstract \( \bar{M}=1.352s \) and realistic \( \bar{M}=1.283s \) commands was 69ms (about 5% of the average trial time). As illustrated in Fig. 6.3a, realistic commands require less
time consistently on average than abstract commands across all blocks, but the difference remains marginal. For training block only, we found a non-significant ($p = .099$) 233ms difference between ABSTRACT ($M=1.794s$) and REALISTIC ($M=1.561s$) conditions.

**Accuracy**

We found a significant main effect of MAPPING on accuracy ($F_{1,165} = 5.32, p = .022, d = 0.23$) but no interaction effect between factors ($p > .05$). Participants were highly accurate in both MAPPINGS, with accuracy for ABSTRACT commands ($M=99.2\%$) higher than that of REALISTIC ($M=98.1\%$) (Fig. 6.3b).

Thus, these results for time and accuracy support that participants are faster when selecting commands from a mapping they are familiar with but slightly become less accurate because of that.

**Retention Rate**

In terms of retention rate, participants could recall the spatial layout for REALISTIC commands ($M=95.1\%$) more effectively than that of ABSTRACT commands ($M=55.7\%$) (Fig. 6.4a) on average. We found significant main effect of MAPPING ($F_{1,11} = 47.19, p < .01, \eta^2_G = .60$) and PHASE ($F_{3,33} = 3.01, p = .044, \eta^2_G = .06$) on retention rate. We also found an interaction effect between MAPPING and PHASE ($F_{3,33} = 4.83, p < .01, \eta^2_G = .09$). Simple main effect analysis revealed that retention for REALISTIC commands was higher than ABSTRACT commands for every PHASE condition (all $p < .001$) and retention for ABSTRACT commands was significantly higher during 0D than during 7D ($p = .047$). Therefore, we are confident that participants have a higher retention rate for mappings they are already familiar with as compared to newly learned mappings.

We further analysed the retention errors and noticed that majority of the errors were incurred when the participants remembered a command to be on a key adjacent to the actual target. For instance, participants recalled ‘South Africa’ [M] key on either the N, J or K keys (Fig. 6.4b). Retention was also easier for targets (e.g. ‘Brazil’ or
Figure 6.4: (a) Retention rate by phase (at the end of the experiment (0D), after 1 day, 3 days, and 7 days) for each mapping. “Correct” rates indicate that the participant recalled the right key, while “Correct+Adjacent” also includes recalls on one of the adjacent keys. Error bars are 95% confidence. (b) Participants tended to misplace shortcuts from the target location (e.g. in green) to one of the keys adjacent to the target (e.g. in orange).

Figure 6.5: Subjective rating for each mapping using a 5-point Likert scale (1: Strongly Disagree, 5: Strongly Agree).

‘South Africa’) with fewer adjacent keys. This suggests that while the retention for these targets is not perfect, participants could still estimate the general area where the command was located even after one week.

Subjective Rating

Participants rated the realistic command set as easier to use and leverage more prior knowledge than the abstract command set. There was a significant main effect of mapping ($V = 3.0, p = .036$) on ease of use, where participants rated realistic (median=4.0) commands more favourably than abstract (median=3.5) commands. There was also a significant main effect of mapping ($V = 4.0, p = .015$) on leveraging prior knowledge, where participants rated realistic (median=4.0) commands more favourably than abstract (median=2.5) commands.
6.9 Discussion

Minor differences between abstract & realistic mappings

Despite the absence of prior knowledge, **ABSTRACT** mapping could still deliver high performance comparable to **REALISTIC** mapping. On average, during our test blocks, participants were only 5% (69ms) slower in **ABSTRACT** mapping than in **REALISTIC** mapping. The time difference is likely due to a shorter visual search, as users were already familiar with the location of the key they were looking for. This is somewhat encouraging as it suggests that participants could be nearly as fast even with new mappings, suggesting that they may switch mapping between applications effortlessly.

Similar learning effects across mapping

In addition, we observed a similar learning effect across the two mapping conditions. As we did not observe any interaction between **BLOCK** and **MAPPINGS**, we can conclude that the time performance gap we observed remained more or less consistent across time. However, more importantly, this lack of interaction also suggests that participants became almost equally fast with SoftCuts no matter which mapping was used, suggesting good potential for users to become expert with the technique rather quickly. Also, participants only required minimal training (one training block) to reach a low selection time. In the end, it is fair to assume that the performance difference measured during our study would fade out once users familiarise with the abstract mapping. We thus believe that SoftCuts could support expert performance regardless of the mapping used.

Role of spatial memory in long-term recall

It was somewhat expected for participants to remember **REALISTIC** command mappings more easily due to years of exposure and practice. However, we were surprised to learn that with only about 15 minutes spent by each participant to complete the 9 blocks of trials in our study, participants could estimate the general command location even after 1 week without rehearsal. Their estimations were spatially accurate, considering that their answers were directly adjacent to the target key being asked to recall. This result is of particular interest, as it means that users would first gaze in the area of the keyboard where this command is located. If we were to include these adjacent key estimation as correct answers, **ABSTRACT** mapping observed retention of close to 70% of the commands after 1 week. This number will only increase with more prolonged and more frequent interaction. Hence, we are confident that SoftCuts not only can be generalised with any mapping across different applications but also accelerate users’ novice-to-expert transition.

6.10 Limitations

One of the challenges for memory recall is when the retrieval of a given piece of information suffer from interference by another similar piece (J. R. Anderson and Bower, 1974). Similar to moded errors (see Section 2.1), Parnas (1969) highlighted how these
could manifest in user errors and overall selection performance. In the context of our hotkeys mapping, the inconsistencies across applications and devices may result in two possible scenarios. It is either 1) the same key mapped to multiple commands or 2) the same command mapped to different keys. For the first scenario, J. Scarr, Cockburn, Gutwin, and Bunt (2012) conducted a study that suggested participants have a strong knowledge of the interface despite having several commands overlapped at the same location. For the second scenario, this challenge applies to virtually any technique, not only SoftCuts. Therefore, future studies need to consider how we could standardise mappings across applications and devices as much as possible.

6.11 Conclusion

This Chapter has explored how command mappings could play a role in maximising user learning. Selections with realistic mapping is only 5% faster as compared to those with abstract mapping. Participants could leverage prior knowledge with a realistic mapping and sustain retention with both mappings even after a week, with the help of spatial memory. This benefits not only novice users but also expert users in the long run, which directly supports the hypothesis we stated in Chapter 1.2.2.
Chapter 7

Comparing Performance between Keyboard and Grid Layout

7.1 Motivation

Another technique has been shown to be fast and accurate on many touch-based devices: FastTap (Gutwin, Cockburn, J. Scarr, et al., 2014). FastTap and SoftCuts are quite different. FastTap, a rehearsal-based interface (RBI), requires a delay, typically of 250ms, before displaying the available commands and does not leverage any prior knowledge of hotkeys. The main advantage of FastTap comes from the full-screen grid layout that takes advantage of large screens, whereas SoftCuts use a space limited to a small keyboard layout. As such, FastTap offers each command larger screen areas. As the performance of a technique is vital for adoption, we thus wondered how this layout difference would impact selection accuracy and speed for SoftCuts. Therefore, comparing a keyboard layout to a full-screen grid-based layout would help to quantify the magnitude of the impact that the choice of relying on a keyboard layout has on performance.

Performance remains an essential point for adopting a technique, as if a technique is too slow, it may never be used (Simon, 1956). However, directly comparing the performance of SoftCuts and FastTap is of little interest and would be highly unfair, given how different these techniques are by nature. First, there is no prior knowledge that users can leverage as easily and clearly as existing hotkeys. Second, FastTap would suffer from a delay penalty for some selections, especially since some studies have shown that users would never stop selecting after this delay (Gutwin, Cockburn, and Lafreniere, 2015; Goguey, Malacria, Cockburn, et al., 2019b) and make errors due to recall issues. However, what is of higher interest is how much SoftCuts compare to a full-screen grid-based layout command mechanism that displays larger targets to quantify the magnitude of the impact that the choice of relying on a keyboard layout has on performance. We believe that since FastTap offers large areas for each command, selection time should be shorter and more accurate than SoftCuts.

7.2 Grid Layout Condition

In order to offer a fair comparison, we implemented a full-screen grid layout similar to FastTap, but we changed a few points detailed below. First, to minimise potential
Chapter 7. Comparing Performance between Keyboard and Grid Layout

7.3 Task

For both grid and keyboard layouts, the target command name would be displayed in the middle of the screen (Fig. 7.1) at the beginning of each trial. Participants then had to select the command via a user-maintained mode, that is, by first holding a modifier key with one finger to reveal the command layout and then pressing the target command key. Note that we decided to rely on a user maintained activation method to remain in line with the FastTap design, but as a reminder, previous work Fennedy, Malacria, et al., 2020 suggests it is a sub-optimal activation method for SoftCuts and possibly for FastTap (but it has not been investigated). However, contrary to FastTap (Gutwin,
Cockburn, J. Scarr, et al., 2014), participants were not forced to perform only chord gestures when selecting a command and could use both hands for both layouts. Upon each selection, participants would receive both audio and visual feedback for 500ms to indicate success or failure. The green colour was used for a correct selection, while the red colour was used for an incorrect selection. The interface would hide the target name as soon as the modifier key has been pressed for a fair comparison between both conditions (grid and keyboard). We instructed each participant to perform the selection task as fast and accurately as possible. For each trial, time was measured from when the target command was first displayed to when the participant invoked a selection.

### 7.4 Stimulus

We used 4 different command mappings (activity, animal, flag, food) of 16 commands (Fig. 7.2). Activity and animal mappings are reserved for the keyboard layout while flag and food mappings are reserved for the grid layout. All 4 mappings contain 4 sub-groups to categorise similar commands together in each row, e.g. for flag mapping, we have diverse countries from Europe, Asia, Africa and North America. This strategy was used by FastTap (Gutwin, Cockburn, J. Scarr, et al., 2014). We similarly applied it to the keyboard layout by having 4 sets of 4 adjacent keys distributed across the 3 rows available on the keyboard layout. This was to ensure that the spatial arrangement of commands between both layouts would be similar and would not result in potential confounding effects on performance. For the keyboard layout, we avoided mnemonic associations between keys and activities/animals, e.g. "P" for Pig. 8 commands out of the total 16 were chosen as targets from each mapping set.
7.5 Procedure

Participants began the experiment by signing a consent form and completing a questionnaire about their demographic information, handedness, type of smartphone, and usage frequency of tablet. Then, each participant had to complete the selection tasks across 4 conditions combining device, layout and mapping variant. In terms of device, we used both a smartphone and a tablet in portrait orientation. For each device, participants had to complete the selection tasks with both layouts, one layout after the other. The mapping used in each condition was not repeated for subsequent conditions belonging to the same participant. This can be done by counterbalancing the mapping associated with the same device and layout combination. For instance, P1 and P2 had the same order for Device and Layout while the mapping was swapped accordingly. For each of the 4 conditions, participants performed 1 training block (B0) followed by 5 test blocks (B1-B5). We ensured that participants did not accumulate fatigue by recommending ample breaks between conditions. Each block consisted of 16 trials, where each of the 8 chosen target commands was repeated twice in random order. We also asked participants to subjectively rate their preferences between both layouts using a Likert Scale after each device condition had been completed. This is so that their perceptions would not suffer from biases across different devices.

7.6 Participants and Apparatus

We recruited 16 participants (8 female, 8 male and all right-handed), aged 19 to 30 (\(M=22.8, \ SD=3.4\)) from the university community. All 16 participants were not involved in previous experiments. 10 of them used an Android smartphone, while 6 of them used iOS smartphone. Only 2 of them had never used any tablet while 5 used daily, 1 used weekly, 4 used monthly and another 4 used yearly. The experimental software was written in Java using Android Studio as an IDE and ran on version 28 of Android SDK. We used a Samsung Galaxy Tab 4 (10.5" screen, 483 grams) running Android 9 for tablet device condition of the experiment and a Samsung Galaxy A10 (6.2" screen, 168 grams) for phone device condition. Each participant received a $5 reimbursement for their 30 minutes participation.

7.7 Design

We used a within-subject design with LAYOUT and DEVICE as primary independent variables and BLOCK as a secondary independent variable. LAYOUT has 2 levels \{GRID, KEYBOARD\}, DEVICE has 2 levels \{PHONE, TABLET\} and BLOCK has 6 levels \{B0-B5\}. The order for device and layout were counterbalanced to minimise any potential order effect. In terms of dependent variables, we measured the time and accuracy of each trial. We measured subjective feedback using a 1-5 Likert Scale (1: strongly disagree, 5: strongly agree) on their overall opinion (“I liked using this technique”) between both layouts, and also in terms of ease of use, speed, accuracy, comfort, and memorability. For the full details of the questions we asked, please refer to Appendix F. The experiment took approximately 30 minutes to complete. In summary, we recorded 16 participants
× 2 techniques × 2 devices × (1 training block + 5 test blocks) × 8 commands × 2 repetitions = 6,144 trials in total.

7.8 Results

We first marked each trial as an outlier if its trial time was above or below 3 standard deviations from the mean time of its respective device and layout condition. A total of 97 trials (1.6%) were removed for subsequent analysis. 34 of which from KEYBOARD-PHONE condition, 15 of which from KEYBOARD-TABLET condition, 27 of which from GRID-PHONE condition, and 21 of which from GRID-TABLET condition. Since time measurement significantly deviated from normality, we applied log transformations. Similarly, we used Aligned Rank Transform (Wobbrock, Findlater, et al., 2011) on accuracy. We used pairwise t-tests with Bonferroni corrections for post hoc comparisons. Trials were aggregated by participant and factors being analyzed. When the assumption of sphericity was violated, we corrected both p-values and degrees of freedom using Greenhouse-Geisser (\(\epsilon < 0.75\)). For subjective rating, we used Wilcoxon signed-rank test for analysis. Additional analysis showed that the icons used in each LAYOUT do not contribute any difference in performance between LAYOUTS.

Learning Effect

There was a significant main effect of BLOCK (\(F_{5,75} = 101.6, p < .0001, \eta^2 = .42\)) on time. Pairwise comparisons revealed that participants required more time during training block B0 and test block B1 (all \(p<.05\)) than the remaining blocks. There was also an interaction effect between LAYOUT, DEVICE, and BLOCK (\(F_{5,75} = 2.45, p = .041, \eta^2 = .01\)). For PHONE condition, KEYBOARD was significantly slower in test block B1 than in B3 (\(p = .026\)) and B5 (\(p = .029\)), while GRID showed no significant difference between B1 and all other blocks. For TABLET condition, both layouts showed that B1 required more time than all other blocks (all \(p<.05\)). These results interestingly suggest that TABLET condition may require more learning than PHONE condition. Therefore, we only removed training block B0 from our subsequent analysis.

Time

We found that participants selected commands faster with GRID than with KEYBOARD. There was a significant main effect of LAYOUT \(F_{1,15} = 38.8, p < .0001, \eta^2 = .19\) with GRID requiring less time than KEYBOARD for both PHONE (219ms or 16.2% faster) and TABLET (164ms or 12.3% faster) conditions (Fig. 7.3b). We did not find any interaction between LAYOUT, DEVICE and BLOCK. We also modelled our interaction in each LAYOUT using (Goguey, Casiez, et al., 2018), and its prediction (that grid layout is more efficient than keyboard layout) match that with ours.

Accuracy

Participants were highly accurate with both LAYOUTS while using each DEVICE, with an average ranging from 97.9% to 98.8% (Fig. 7.4b). We found no significant main effect of LAYOUT (\(p = .79\)) or DEVICE (\(p = .54\)) on accuracy, and no interaction. These results
Chapter 7. Comparing Performance between Keyboard and Grid Layout

Figure 7.3: (a) Time by block for each layout and device. (b) Time by device for each layout while excluding blocks that demonstrated learning effect. Error bars are 95% confidence.

Figure 7.4: (a) Accuracy by block for each layout and device. (b) Accuracy by device for each layout. Error bars are 95% confidence.
suggest that the accuracy of both layouts are similarly high, although the KEYBOARD layout occupies less than a third of the space occupied by the GRID.

**Subjective Rating**

Participants rated that using GRID is easier, more accurate and more comfortable than using KEYBOARD while on PHONE but perceived both layouts to be similar while on TABLET (Fig. 7.5). For phone, there was a significant main effect of LAYOUT ($V = 83.5, p = .048$) on ease of use, where participants rated GRID (median=4.5) more favourably than KEYBOARD (median=4.0). There was another significant main effect of LAYOUT ($V = 80.0, p = .014$) on accuracy, where participants rated GRID (median=5.0) more favourably than KEYBOARD (median=3.5). There was another significant main effect of LAYOUT ($V = 79.0, p = .014$) on comfort, where participants rated GRID (median=5.0) more favourably than KEYBOARD (median=4.0). For tablet, we did not find any significant main effect of LAYOUT on any measure. Overall, 11 out of 16 participants chose a GRID layout over a KEYBOARD as their preferred LAYOUT for each DEVICE.

### 7.9 Discussion

**Limited speed benefit of a Grid layout**

Our results revealed a clear performance gap between the speed of a grid layout (160-220 ms faster) and a keyboard. This was expected when one considers that this advantage comes at the cost of full-screen occlusion, making the pointing task of selecting a command easier for the grid layout. While Gutwin, Cockburn, J. Scarr, et al. (2014) suggested using partial transparency to alleviate this problem, the visual conflict between actual content and command layout may not be easy to ignore, posing a serious challenge to eventual adoption. More recently, work from Henderson, Malacria, et al. (2020) suggested that visual occlusion might not be a problem with Marking Menus (another type of RBI). However, it is unclear whether these results would apply to full-screen interfaces like FastTap Henderson, Malacria, et al., 2020.

It is important to note that the grid layout we implemented did not use the advocated delay (of 250 ms) before displaying the grid layout, potentially reducing the
performance gap between SoftCuts and the actual FastTap. Adding this delay would reduce and might even negate the time advantage of FastTap over SoftCuts. This delay would also be a long term problem for users: research conducted specifically on the adoption of expert mode with FastTap revealed difficulties to adopt this mode (Goguey, Malacria, Cockburn, et al., 2019b; Gutwin, Cockburn, and Lafreniere, 2015; Lafreniere, Gutwin, and Cockburn, 2017), suggesting that this delay will always impact a decent proportion of commands. In addition, the performance of the keyboard layout could be improved by using a faster input method such as Once (see Chapter 5), and spreading the shortcuts instead of using 4 groups of 4 adjacent commands. Hence, despite its slightly higher command selection time, a keyboard layout presents a desirable alternative, especially when each of its keys consumes 88-90% less screen real estate for a 12-16% speed cost than the grid layout. Finally, it is worth noting that there was no benefit of the grid layout over a keyboard layout in terms of accuracy.

Smaller screen estate is an advantage for the keyboard layout

To our surprise, the keyboard layout did not suffer from any accuracy drop despite having a much smaller margin of error (reported by P2,6,9) due to the smaller keyboard layout. In fact, both layouts similarly demonstrated a high accuracy range between 97-99% for both phones and tablets. Although our subjective rating showed that participants rated the grid layout as more accurate and comfortable to use than the keyboard layout, we found conflicting remarks by some participants when we asked them to elaborate on their reasons. They shared that “having the keys distributed over a larger surface area [with the grid layout] means that my fingers had to move over a bigger distance” [P2,3,5,8]. This reachability issue was not observed when using the keyboard layout because participants “could use minimal wrist movement and hence require less effort” [P3,7,15] to reach all corners of the keyboard. The keyboard layout induced a bimanual posture of interaction that was found to be “more comfortable” [P1,2] and allowed P9, P14, and P16 to support the weight of the tablet by “anchoring [their] thumb” [P9]. Therefore, the keyboard offers a clear advantage in terms of comfort as compared to the grid layout for implementations across multiple platforms.

Keyboard layout supports greater spatial stability

While both layouts incorporate spatial stability in their designs, we have to acknowledge the challenge of integrating each layout across different applications and device sizes. The grid-based strategy does not have to follow a standardised mapping, unlike our keyboard-based strategy. For instance, a grid layout would require more effort to ensure that designers and developers from different applications choose a consistent grid size and position within the grid for the same command implemented. Also, Gutwin, Cockburn, J. Scarr, et al. (2014) suggested that it is possible to “have a grid of up to 40 buttons in an 8x5 layout on a 7-inch tablet”. This flexibility may result in greater spatial instability of commands across devices and reduced performance if the individual area for each command becomes smaller. On the other hand, SoftCuts use a keyboard layout that ensures a useful constraint of 26 keys maximum (assuming each character key represents a command) regardless of the device being used. Spatial stability is also better preserved in our current implementation of SoftCuts because it
can adopt the mappings that have been mostly standardised for physical keyboards for decades.

7.10 Limitations

We acknowledge that actual world context of text-editing or sketching application may impose additional constraints, as compared to what was introduced in our study. Future studies should consider how different screen space occupied by the interface may impact user preference and actual task performance.

7.11 Conclusion

This chapter compares SoftCuts’ keyboard layout with FastTap’s grid layout to find out which layout supports user performance. Results show that grid layout is faster by 12-16% while accuracy is similarly high despite keyboard layout’s smaller screen real estate. We also show how keyboard layout supports greater spatial stability than grid layout. This valuable insight demonstrates why SoftCuts offer a competitive advantage against an alternative approach to efficient command selections.
Chapter 8

Discussion and Future Work

8.1 Expanding input vocabulary for SoftCuts

![Figure 8.1](image)

**Figure 8.1:** (a) Example of SoftCuts on an Android soft keyboard with a `ctrl` modifier key. (b) Tapping `ctrl` once displays the first level of shortcuts (e.g., `ctrl`+`s` for save), and substitutes `#1` with `alt`. (c) Tapping on `alt` displays additional shortcuts that are usually triggered with both `ctrl` and `alt` modifier keys.

We have only investigated the feasibility and benefits of using one modifier key (e.g., `ctrl`). However, hotkeys in general may utilise additional modifier keys like `alt` (Windows) or `option` (Mac OS). As `alt` is already present on every existing soft keyboard, it would be relatively simple to add support for commands relying on it (e.g., `ctrl`+`U`+`Z` for redo). Existing layouts could also be adapted to add more modifier keys (Figure 8.1). The changes to the layout would be minimal and could follow one of the two following strategies:

1. Reduce the size of other keys (e.g., `space`) to add the additional modifier key(s) to the existing layout.

2. Replace a key that is not used in command selection mode (i.e., when SoftCuts is active) and affect the layout in that mode.

Using the first option potentially makes it easier for users to discover SoftCuts and suggest using the keys. However, it could interfere with typing tasks and thus impact

---

1 Some applications implement `ctrl`+`Y` or `ctrl`+`Z` as hotkeys for redo command.
their performance. Replacing a key (e.g. from $\#1$ to $\text{alt}$) (see Fig. 8.1b) would keep the layout consistent, but it is unclear whether users would notice the subtle change. Future work needs to investigate the ideal feedforward mechanism to suggest additional shortcuts to be used on a given key. For example, pressing on $Z$ alone would invoke the undo command, but users may not be aware that there is a possibility of invoking a redo command using $\text{alt} + 1 + Z$. Finally, supporting more modifier keys could also limit the use of the UM input method, as maintaining three or more keys pressed at once may be challenging if not impossible on touch-based devices. Therefore, an alternative approach is to consider limiting the input method to only Once and Swipe regardless if the number of modifier key chording supported is one or more than one.

8.2 Standardisation of hotkey mappings

The fundamental building block of SoftCuts lies in the establishment of a standardised hotkey mapping. Without it, SoftCuts would degenerate into the current command selection mechanism, which faces limitation in terms of discoverability, consistency and efficiency. While many basic commands like cut, copy, and paste have been implemented without much discrepancies across today’s applications and devices, there is room for improvement for other more advanced commands. For instance, taking a screenshot on a Windows desktops require chording between $\text{alt}$ and $\text{PrtScn}$ (an abbreviation for print screen command) keys, while MacOS systems would require users to chord $\text{alt}$, $1$, and $3$ keys. These discrepancies limit the full potential of SoftCuts.

We propose the following steps for individuals and organisations to consider taking in the near future.

1. Assemble a consortium representing relevant stakeholders, from software/hardware companies (e.g. Apple, Google, and Microsoft) to research institutions, and to government ministries.

2. Consolidate existing hotkeys mapping implemented across applications and devices. Defkey\(^2\)’s comprehensive database can serve as a starting point or reference.

3. Review inconsistencies by using telemetry data to better understand users’ actual hotkey usage frequency for a specific command or conducting studies to better understand ergonomic preference and feasibility of chording certain key combinations. Consortium members can then make informed decisions through discussions and voting process.

4. Publish and promote a universally accessible standard.

Despite decades of witnessing unregulated implementation hotkey mappings, it is better late than never to kick-start these much-needed mapping standards. In fact, our above steps are inspired by the successful Unicode Standard\(^3\) developed by the Unicode Consortium to provide a unique number for every character regardless of platform, program, or language. The emergence of the Unicode Standard has benefited the

\(^{2}\)https://defkey.com/
\(^{3}\)https://home.unicode.org/
processing, storage and interchange of text data in all modern software and information technology protocols.

In a similar vein, we believe that establishing a standard for hotkeys mapping could pave the way for designers and developers to rely on a practical guide while enhancing existing mapping or adopting hotkeys in future applications. Standardisation will not only solve the mapping interference problem (as mentioned in Chapter 6.10), but also indirectly increase the performance ceiling of both novices and experts, once adopted by major computer hardware companies.

### 8.3 Reversing Skill Transfer

While participants could leverage prior knowledge of hotkeys from desktop to mobile platforms using SoftCuts (see Chapter 6), we believe that a similar skill transfer could happen in the opposite direction. This is supported by evidence from the latest worldwide market share (see 8.2) which highlights how the sales of mobile devices exceeded that of desktop between 2016 and 2017. This trend will only become more pronounced over the next decade, especially with the advent of 5G network technology which will revolutionise the way users interact with their mobile devices. Hence, if SoftCuts were to be integrated into mainstream mobile devices, this means that the users may be exposed to SoftCuts before the traditional physical hotkeys.

![Figure 8.2: Worldwide market share of desktop vs. smartphone vs. tablet. Data source: https://gs.statcounter.com/](image-url)
Using SoftCuts as a unified command selection mechanism on mobile devices could thus teach users about hotkeys in general and may allow them to leverage any knowledge of SoftCuts on desktop computers. SoftCuts users could potentially identify a familiar modifier key (\textasciitilde{\text{ctrl}}) and then be tempted to press it on a desktop application. If that first step was to be completed, computers could use visual help to show possible shortcuts. Notably, we believe that KeyMap (Lewis, d’Eon, et al., 2020), which uses a very similar visual representation, by showing a keyboard on the bottom part of a desktop computer screen (see Fig. 2.14d), could augment users’ familiarity with SoftCuts and help them get familiar with regular hotkeys.

8.4 Adaptation for Extended Reality (XR) Environment

Keyboards may predominantly be used to support interaction on 2D surfaces like desktop and mobile devices. On the other hand, the growing 3D environments like Mixed Reality (MR) and Virtual Reality (VR) have witnessed the benefits of relying on a keyboard to augment its primary mid-air interaction. For instance, Speicher et al. (2018) have shared the empirical advantage of pointing using tracked hand-held controllers with a virtual keyboard (see Fig. 8.3a). However, it is the physical keyboard that serves as a promising input device due to its tactility. Most users are already familiar with the consistent QWERTY keyboard layout, so much so that they could blind-type (McGill et al., 2015). This is why recent studies have been exploiting the ubiquity of physical keyboards to support text entry in VR. Grubert, Witzani, et al. (2018) demonstrated how novice users could retain 60% of their typing speed on physical keyboard (see Fig. 8.3b), 15% higher than a soft keyboard. Typing performance can also be enhanced (Knierim et al., 2018) by tracking actual hands’ movements and visualising them as.
Figure 8.4: Proposed design for SoftCuts in AR or MR environment.

semi-transparent virtual hands (see Fig. 8.3c). Therefore, it is safe to assume that the keyboard is going to stay despite the change in the interactive environment from 2D to 3D. This is especially true when the office of the future will be redesigned with virtual interfaces (Grubert, Ofek, et al., 2018; Mcgill et al., 2020) to boost productivity anytime and anywhere.

While our Chapter 2 covers strategies and theories focusing on 2D interfaces, some of them translate to a 3D environment to a certain extent. For instance, Surale, Matulic, and Vogel (2019) extended its mode-switching evaluation to include bare-handed mid-air techniques. They shared that most speed performance is comparable with that of touch-based. Perrault et al. (2015) demonstrated that Physical Loci could outperform mid-air MM in terms of retention by creating associations between physical objects in a room, and this demonstrates the benefits of 3D spatial memory. D. Schneider et al. (2019) has attempted to reconfigure the input and output space of individual keys of a physical keyboard in VR. This is similar to that of Optimus Keyboard (see Fig. 8.3) highlighted in Section 2.2.1. However, they do not exploit the fact that users could benefit from their prior knowledge by using hotkeys’ established mapping. Therefore, the evidence is clear that there are untapped opportunities to extend SoftCuts in such an immersive environment.

We imagine how hotkeys could be extended to support command selections in both AR/MR and VR. For AR/MR, a virtual layer of keyboard layout can be overlaid on top of the tracked physical keyboard when the user presses the modifier key (see Fig. 8.4). Similarly, if a different modifier key is pressed instead, then the rendering should be updated dynamically according to the active context triggered by the user. For VR, we consider a scenario where the user still sticks with the old habit of

5https://tech.fb.com/the-future-of-work-and-the-next-computing-platform/
Figure 8.5: Proposed design for SoftCuts in VR environment.
invoking commands through menus. We can print the keyboard layout adjacent to the menu item selection (see Fig. 8.5 to motivate users to switch to expert usage of hotkeys. However, we have not validated the effectiveness of these ideas and encourage future studies to investigate how one design would perform compared to others.
Chapter 9

Conclusion

The multitouch surface interaction in the past decade has made it possible for more users to benefit from technology anywhere and anytime. Thus, mobile devices equipped with this interactivity, such as smartphones and tablets, have penetrated increasing aspects of our everyday lives in a remarkably short period of time, and this trend is likely to continue in the coming years. The nature of the simple, direct and intuitive touch actions may be the main reason for such improved usability over the conventional desktop interaction with keyboard and mouse. The other side of the coin is that the limited gesture vocabulary employed makes the transition from a desktop application to a multitouch application difficult. This is especially true for feature-rich applications like Adobe Photoshop, which faces the challenge of accommodating the entire library of commands within the smaller screen size constraint of smartphones and tablet.

As a result, what we witness in today’s landscape of mobile devices is that either a much-reduced functionality of the application is released, or users are left to struggle with discovering the commands. For instance, sometimes, users have to traverse menu hierarchies to select that one desired command (see Fig. 2.8). Alternatively, gestures like shaking the phone, directional strokes or chording (see Fig. 2.11 and 2.12) are mostly hidden, and thus, only a small group of users could use it, and that is if they could remember how. This poor discoverability of command is aggravated by the inconsistency in the implementation of command mapping across applications and OSes. Users ended up having to relearn which input maps to which command whenever they switch between applications or OSes, which often consider the need for multitasking to support productivity on the go.

This is why there is a real and urgent need to unify command selection mechanisms on mobile devices. One simple solution we think could be the answer is to capitalise the ubiquitous yet untapped potential of soft keyboard as not only to support text-entry but also command-selections. We name this mechanism as SoftCuts, which stands for soft keyboard shortcuts. The consistent keyboard layout has been established and used for decades on a desktop computer setup that relies on keyboard and mouse interaction. This thesis explores design options to maximise its benefits when adapting for the multitouch keyboard scenarios.

We conducted a total of 7 studies across 5 different projects to evaluate our hypothesis: can a keyboard layout offer a unified and efficient command selection mechanism on mobile devices for both novice and expert users? As illustrated in Fig. 1.5, the details
Chapter 9. Conclusion

Figure 9.1: Web browsing scenario of soft keyboard shortcuts using Once method. (a) a semi-transparent modifier key is available in the browser, (b) when user taps it, all available shortcuts will be presented to facilitate browsing, (c) user then taps on f (for ‘find’ command) and hence, (d) every occurrence of the “HCI” word (as typed by the user) is highlighted in yellow.
of each project have been presented as an individual chapter of the thesis. In Chapter 3, we start by redesigning soft keyboards to optimise the mode-switching performance while typing complex sets of characters. The design process helped us identify an opportunity to introduce a modifier key like [⌘] on smartphones, making it possible to support command selection using our soft keyboards. Subsequently, we focus on designing essential visual elements of the keyboard (i.e. individual letter keys and modifier key) in Chapter 4 to ensure that novice users could easily discover SoftCuts and know how to use it without external help. Aside from optimising usability, it is equally important to study the empirical performance of SoftCuts in delivering efficient yet accurate command selections. Chapter 5 investigates the performance and adoption of input methods, while Chapter 6 investigates how command mappings could affect performance. These insights are then complemented with Chapter 7, which benchmark SoftCuts’ keyboard layout strategy against the state-of-the-art grid layout. Integrating the insights from our studies, Fig. 9.1 depicts how we imagine how the concept of SoftCuts could be easily incorporated into existing applications. All in all, we have demonstrated and hopefully convinced the readers of this thesis that SoftCuts could deliver a robust command selection performance on mobile devices regardless of expertise in hotkeys.

Last but not least, our ideal vision for SoftCuts is that it could inspire existing or future HCI researchers to consider the implications of their designs beyond a singular target scenario. What works best for a particular demographic of users on a specific device may suffer when translated to a slightly different or completely unexpected use case. This thesis investigates how SoftCuts could work for different device configurations under varying mobility conditions while simultaneously paying attention to both novice and expert users’ needs, especially smooth the transition between them. We are only at the beginning of what seems to be a possible alternative future where a unified interaction mechanism could and should play an active role in designing more usable complex applications.
Appendix A

Questionnaire: Visual Representation of Commands

This appendix includes the full questionnaire that was used in Section 4.2.

1. Age: _____

2. Gender:
   □ Male □ Female □ Non-binary □ Not listed □ Prefer not to say

3. Handedness: □ Left □ Right □ No preference

4. How often do you use keyboard shortcuts on desktop/laptop?
   □ 1 (never)
   □ 2
   □ 3 (sometimes)
   □ 4
   □ 5 (always)

5. (Notes App)
   For each of the following statement, please give a score from 1 (strongly disagree) to 7 (strongly agree).

   • I like using this design.
     D1: _____  D2: _____  D3: _____

   • I find this design easy to use.
     D1: _____  D2: _____  D3: _____

   • I find this design fast to use.
     D1: _____  D2: _____  D3: _____

   • I find this design accurate to use.
     D1: _____  D2: _____  D3: _____

   • I find this design comfortable to use.
     D1: _____  D2: _____  D3: _____

   • I find this design useful.
     D1: _____  D2: _____  D3: _____

   • I find this design visually appealing.
     D1: _____  D2: _____  D3: _____
6. (Notes App)
What is your overall design preference?
☐ D1
☐ D2
☐ D3

7. (Browser App)
For each of the following statements, please give a score from 1 (strongly disagree) to 7 (strongly agree).

- I like using this design.
  D1: _____ D2: _____ D3: _____

- I find this design easy to use.
  D1: _____ D2: _____ D3: _____

- I find this design fast to use.
  D1: _____ D2: _____ D3: _____

- I find this design accurate to use.
  D1: _____ D2: _____ D3: _____

- I find this design comfortable to use.
  D1: _____ D2: _____ D3: _____

- I find this design useful.
  D1: _____ D2: _____ D3: _____

- I find this design visually appealing.
  D1: _____ D2: _____ D3: _____

8. (Browser App)
What is your overall design preference?
☐ D1
☐ D2
☐ D3
Appendix B

Questionnaire: Familiarity and Saliency of Modifier Keys

This appendix includes the full questionnaire that was used in Section 4.3.

1. Age: _____

2. Gender:
   □ Male □ Female □ Non-binary □ Not listed □ Prefer not to say

3. How many years of experience do you have with a smartphone?
   □ 0 to 2 years
   □ 2 to 5 years
   □ 5 to 15 years
   □ more than 15 years

4. How many years of experience do you have with a desktop or laptop computer?
   □ 0 to 2 years
   □ 2 to 5 years
   □ 5 to 15 years
   □ more than 15 years

5. Which desktop OS are you most familiar with?
   □ macOS
   □ Windows
   □ Linux

6. For each of the following statement, please give a score from 1 (strongly disagree) to 7 (strongly agree).
   • I find SoftCuts useful. _____
   • I find SoftCuts easy to use. _____
   • I like typing on my phone. _____
     Please ignore the above statement and intentionally choose ’3’ as an answer.
   • I find SoftCuts easy to discover. _____
   • I find SoftCuts easy to recall keyboard shortcuts. _____
   • I find SoftCuts visually appealing. _____
• I would like to see SoftCuts available on my smartphone and/or tablet device.

7. These are 3 design variants for SoftCuts. Which one do you prefer most?
   - D1
   - D2
   - D3

8. Do you have any thoughts, feedback, or ideas to share about SoftCuts?

9. How regularly do you use keyboard shortcuts on your computer?
   - 1 (never)
   - 2
   - 3
   - 4 (sometimes)
   - 5
   - 6
   - 7 (always)

10. How many keyboard shortcuts do you know/use?
    - 1 to 5
    - 6 to 10
    - 11 to 20
    - 21 to 30
    - more than 30

11. How many years of experience do you have with keyboard shortcuts?
    - 0 to 2 years
    - 2 to 5 years
    - 5 to 15 years
    - more than 15 years

12. Any comment/question for us?
Appendix C

Questionnaire: Performance of Input Methods

This appendix includes the full questionnaire that was used in Section 5.2.

1. Age: _____

2. Gender:
   □ Male □ Female □ Non-binary □ Not listed □ Prefer not to say

3. Which smartphone OS do you use?
   □ android
   □ iOS
   □ Windows

4. How regularly do you use a tablet?
   □ once a day
   □ once a week
   □ once a month
   □ once a year
   □ never

5. Do you use a Swype-type keyboard?
   □ Yes
   □ No

6. (Posture #1: Phone 1H Portrait)
   For each of the following statement, please give a score from 1 (strongly disagree) to 5 (strongly agree).

   • I like using this technique.
     Once: _____  Swipe: ____

   • I find this technique easy to use.
     Once: _____  Swipe: ____

   • I find this technique comfortable to use.
     Once: _____  Swipe: ____

   • I find this technique fast to use.
     Once: _____  Swipe: ____
• I find this technique accurate to use.
  Once: ______  Swipe: ______

7. (Posture #1: Phone 1H Portrait)
What is your overall technique preference?
☐ UM
☐ Once
☐ Swipe

8. (Posture #2: Phone 2H Portrait)
For each of the following statement, please give a score from 1 (strongly disagree) to 5 (strongly agree).

• I like using this technique.
  UM: ______  Once: ______  Swipe: ______
• I find this technique easy to use.
  UM: ______  Once: ______  Swipe: ______
• I find this technique comfortable to use.
  UM: ______  Once: ______  Swipe: ______
• I find this technique fast to use.
  UM: ______  Once: ______  Swipe: ______
• I find this technique accurate to use.
  UM: ______  Once: ______  Swipe: ______

9. (Posture #2: Phone 2H Portrait)
What is your overall technique preference?
☐ UM
☐ Once
☐ Swipe

10. (Posture #3: Phone 2H Landscape)
For each of the following statement, please give a score from 1 (strongly disagree) to 5 (strongly agree).

• I like using this technique.
  UM: ______  Once: ______  Swipe: ______
• I find this technique easy to use.
  UM: ______  Once: ______  Swipe: ______
• I find this technique comfortable to use.
  UM: ______  Once: ______  Swipe: ______
• I find this technique fast to use.
  UM: ______  Once: ______  Swipe: ______
• I find this technique accurate to use.
  UM: ______  Once: ______  Swipe: ______
11. (Posture #3: Phone 2H Landscape)
   What is your overall technique preference?
   □ UM
   □ Once
   □ Swipe

12. (Posture #4: Tablet 2H Portrait)
   For each of the following statement, please give a score from 1 (strongly disagree) to 5 (strongly agree).
   - I like using this technique.
     UM: _____  Once: _____  Swipe: _____
   - I find this technique easy to use.
     UM: _____  Once: _____  Swipe: _____
   - I find this technique comfortable to use.
     UM: _____  Once: _____  Swipe: _____
   - I find this technique fast to use.
     UM: _____  Once: _____  Swipe: _____
   - I find this technique accurate to use.
     UM: _____  Once: _____  Swipe: _____

13. (Posture #4: Tablet 2H Portrait)
   What is your overall technique preference?
   □ UM
   □ Once
   □ Swipe

14. (Posture #5: Tablet 2H Landscape)
   For each of the following statement, please give a score from 1 (strongly disagree) to 5 (strongly agree).
   - I like using this technique.
     UM: _____  Once: _____  Swipe: _____
   - I find this technique easy to use.
     UM: _____  Once: _____  Swipe: _____
   - I find this technique comfortable to use.
     UM: _____  Once: _____  Swipe: _____
   - I find this technique fast to use.
     UM: _____  Once: _____  Swipe: _____
   - I find this technique accurate to use.
     UM: _____  Once: _____  Swipe: _____

15. (Posture #5: Tablet 2H Landscape)
   What is your overall technique preference?
   □ UM
   □ Once
   □ Swipe
Appendix D

Questionnaire: Adoption of Input Methods

This appendix includes the full questionnaire that was used in Section 5.3.

1. Age: _____

2. Gender:
   - [ ] Male  [ ] Female  [ ] Non-binary  [ ] Not listed  [ ] Prefer not to say

3. Handedness:  [ ] Left  [ ] Right  [ ] No preference

4. Which smartphone OS do you use?
   - [ ] android
   - [ ] iOS
   - [ ] Windows

5. How regularly do you use a tablet?
   - [ ] once a day
   - [ ] once a week
   - [ ] once a month
   - [ ] once a year
   - [ ] never

6. For each of the following condition, What is your overall technique preference?

   • Sitting Phone 1H Portrait
     Once: _____  Swipe: _____

   • Sitting Phone 2H Portrait
     UM: _____  Once: _____  Swipe: _____

   • Sitting Tablet 2H Portrait
     UM: _____  Once: _____  Swipe: _____

   • Sitting Tablet 2H Landscape
     UM: _____  Once: _____  Swipe: _____

   • Standing Phone 1H Portrait
     Once: _____  Swipe: _____

   • Standing Phone 2H Portrait
     UM: _____  Once: _____  Swipe: _____
Appendix D. Questionnaire: Adoption of Input Methods

• Standing Tablet 2H Portrait
  UM: _____  Once: _____  Swipe: _____

• Standing Tablet 2H Landscape
  Once: _____  Swipe: _____

• Walking Phone 1H Portrait
  Once: _____  Swipe: _____

• Walking Phone 2H Portrait
  UM: _____  Once: _____  Swipe: _____

• Walking Tablet 2H Portrait
  UM: _____  Once: _____  Swipe: _____

• Walking Tablet 2H Landscape
  UM: _____  Once: _____  Swipe: _____
Appendix E

Questionnaire:

This appendix includes the full questionnaire that was used in Chapter 6.

1. Age: _____

2. Gender:
   □ Male □ Female □ Non-binary □ Not listed □ Prefer not to say

3. Handedness: □ Left □ Right □ No preference

4. Which smartphone OS do you use?
   □ android
   □ iOS
   □ Windows

5. How regularly do you use a tablet?
   □ once a day
   □ once a week
   □ once a month
   □ once a year
   □ never

6. Have you used Microsoft Word? □ Yes □ No

7. Pre-Experiment Shortcuts Knowledge.
   Which of the following functions are you aware of the corresponding shortcut keys?
   □ Close □ Redo □ Underline □ Italicise □ Print □ Select All □ Save □ Font □ Find □
   Add Link □ Undo □ Cut □ Copy □ Paste □ Bold □ New

8. For each of the following statement, please give a score from 1 (strongly disagree) to 5 (strongly agree).
   
   • I could complete the task easily.
     Countries: _____   Text-Editing: _____
   
   • I could complete the task fast.
     Countries: _____   Text-Editing: _____
   
   • I could complete the task accurately.
     Countries: _____   Text-Editing: _____
   
   • I was able to leverage my prior knowledge of hotkeys for this mapping.
     Countries: _____   Text-Editing: _____
Appendix E. Questionnaire:

9. Which task do you prefer to complete overall?
   - Countries
   - Text-Editing

10. Post-Experiment Shortcuts Knowledge.
    Which of the following functions are you aware of the corresponding shortcut keys?
    - Close
    - Redo
    - Underline
    - Italicise
    - Print
    - Select All
    - Save
    - Font
    - Find
    - Add Link
    - Undo
    - Cut
    - Copy
    - Paste
    - Bold
    - New

11. (0 day away from the first experiment)
    What are the corresponding key for each of the following item:

    - Paste: _____
    - Find: _____
    - Print: _____
    - Undo: _____
    - Save: _____
    - Underline: _____
    - Cut: _____
    - Add Link: _____
    - Vietnam: _____
    - Brazil: _____
    - South Africa: _____
    - Iran: _____
    - Germany: _____
    - Canada: _____
    - Australia: _____
    - Denmark: _____
Appendix E. Questionnaire:

12. (1 days away from the first experiment)
What are the corresponding key for each of the following item:

- Paste: _____
- Find: _____
- Print: _____
- Undo: _____
- Save: _____
- Underline: _____
- Cut: _____
- Add Link: _____
- Vietnam: _____
- Brazil: _____
- South Africa: _____
- Iran: _____
- Germany: _____
- Canada: _____
- Australia: _____
- Denmark: _____

13. (3 days away from the first experiment)
What are the corresponding key for each of the following item:

- Paste: _____
- Find: _____
- Print: _____
- Undo: _____
- Save: _____
- Underline: _____
- Cut: _____
- Add Link: _____
- Vietnam: _____
- Brazil: _____
- South Africa: _____
- Iran: _____
- Germany: _____
- Canada: _____
- Australia: _____
• Denmark: _____

14. (7 days away from the first experiment)
What are the corresponding key for each of the following item:

• Paste: _____
• Find: _____
• Print: _____
• Undo: _____
• Save: _____
• Underline: _____
• Cut: _____
• Add Link: _____
• Vietnam: _____
• Brazil: _____
• South Africa: _____
• Iran: _____
• Germany: _____
• Canada: _____
• Australia: _____
• Denmark: _____
Appendix F

Questionnaire:

This appendix includes the full questionnaire that was used in Chapter 7.

1. Age: ______

2. Gender:
   - [ ] Male  [ ] Female  [ ] Non-binary  [ ] Not listed  [ ] Prefer not to say

3. Handedness:  [ ] Left  [ ] Right  [ ] No preference

4. Which smartphone OS do you use?
   - [ ] android
   - [ ] iOS
   - [ ] Windows

5. How regularly do you use a tablet?
   - [ ] once a day
   - [ ] once a week
   - [ ] once a month
   - [ ] once a year
   - [ ] never

6. (Phone)
   For each of the following statement, please give a score from 1 (strongly disagree) to 5 (strongly agree).

   • I find this layout easy to use.
     Keyboard: _____  Grid: _____

   • I find this layout fast to use.
     Keyboard: _____  Grid: _____

   • I find this layout accurate to use.
     Keyboard: _____  Grid: _____

   • I find this layout comfortable to use.
     Keyboard: _____  Grid: _____

   • I find this layout helps to memorise command location easily.
     Keyboard: _____  Grid: _____

7. (Phone)
   What is your overall layout preference?
Appendix F. Questionnaire:

☐ Keyboard
☐ Grid

8. (Tablet)
For each of the following statement, please give a score from 1 (strongly disagree) to 5 (strongly agree).

• I find this layout easy to use.
  Keyboard: _____  Grid: _____

• I find this layout fast to use.
  Keyboard: _____  Grid: _____

• I find this layout accurate to use.
  Keyboard: _____  Grid: _____

• I find this layout comfortable to use.
  Keyboard: _____  Grid: _____

• I find this layout helps to memorise command location easily.
  Keyboard: _____  Grid: _____

9. (Tablet)
What is your overall layout preference?
☐ Keyboard
☐ Grid
Bibliography


Fitts, Paul M and Michael I Posner (1967). “Human performance.” In:
Bibliography


Isenberg, Tobias and Mark Hancock (May 2012). “Gestures vs. Postures: ‘Gestural’ Touch Interaction in 3D Environments”. In: *Proceedings of the CHI Workshop on “The 3rd Dimension of CHI: Touching and Designing 3D User Interfaces”* (3DCHI 2012, May 5, 2012, Austin, TX, USA). Ed. by Ken Anderson et al. published online, United States, pp. 53–61. URL: https://hal.inria.fr/hal-00781237.


— (May 2010). “Natural User Interfaces Are Not Natural”. In: Interactions 17.5, pp. 46–49. ISSN: 1072-5520. DOI: 10.1145/1744161.1744163. URL: https://doi.org/10.1145/1744161.1744163.


Scarr, Joseph Laurence (2014). “Understanding and Exploiting Spatial Memory in the Design of Efficient Command Selection Interfaces”. In:


