Finger-Aware Shortcuts

Jingjie Zheng, Daniel Vogel
Cheriton School of Computer Science, University of Waterloo
Waterloo, Canada
{ j49zheng, dvogel } @uwaterloo.ca

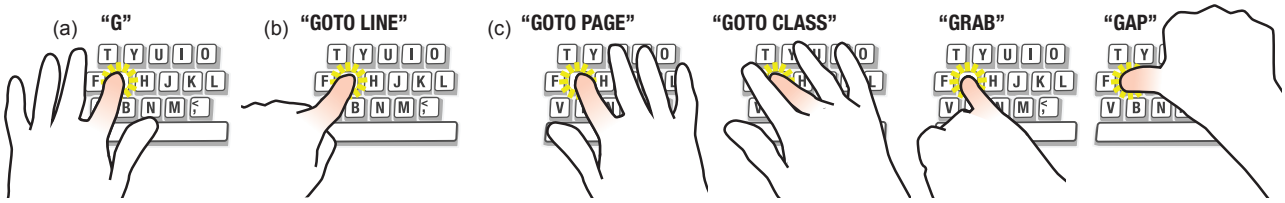

Figure 1: Finger-Aware Shortcuts trigger different commands by detecting the finger, hand, and posture used to press the key: (a) for example, pressing G with the left index finger and open hand simply enters 'G'; (b) pressing G with the left index and closed hand could trigger a command like 'goto line'; and (c) pressing G with different right-hand fingers and postures could trigger other commands like 'goto page', 'goto class', etc.

ABSTRACT

We evaluate and demonstrate finger, hand, and posture identification as keyboard shortcuts. By detecting the hand and finger used to press a key, and open or closed hand postures, a key press can have multiple command mappings. A formative study reveals performance and preference patterns when using different fingers and postures to press a key. The results are used to develop a computer vision algorithm to identify fingers and hands on a keyboard captured by a built-in laptop camera and reflector. This algorithm is built into a background service to enable system-wide finger-aware shortcut keys in any application. A controlled experiment uses the service to compare the performance of Finger-Aware Shortcuts with existing methods. The results show Finger-Aware Shortcuts are comparable with a common class of shortcuts using multiple modifier keys. Finally, application demonstrations illustrate different use cases and mappings for Finger-Aware Shortcuts and extend the idea to two-handed key presses, continuous parameter control, and menu selection.

Author Keywords

keyboard shortcuts; finger identification;

ACM Classification Keywords

H.5.2. Information Interfaces (e.g., HCI): Input devices

INTRODUCTION

Physical keyboards were originally designed for text-entry, but pressing keys can also issue commands with *keyboard shortcuts* (also called "hotkeys" [9]). Shortcut keys repurpose standard text entry keys and may be differentiated from dedicated function keys like F1 and HOME. In text-heavy applications, all shortcut keys need to include one or more special modifier keys (e.g. command ⌘, control ctrl).

Keyboard shortcuts may be difficult to learn [16], but once mastered they are fast [16, 11]. Although commonly used by experts for applications like video editing [10] and programmers like Donald Knuth [13], studies show keyboard shortcuts are underused by most computer users [16, 25, 29]. Researchers attribute this to a gulf between graphical input and pressing keys [15, 22], poor visibility and mnemonics [9, 6], and the uncomfortable and error-prone act of pressing multiple keys simultaneously [21]. Several techniques have been proposed to foster shortcut key usage with interventions and visualizations for training and encouragement [14, 9, 18, 29] as well as augmenting the keyboard or mouse to make shortcut keys more available [21, 26] and expressive [1].

Our work relates to shortcut availability and expressivity by making shortcut keys "finger-aware." We detect which hand and finger is used to press a key, and whether the hand posture is open or closed, so that pressing the same key can have multiple command mappings. This enables a larger input space for traditional keyboards with increased expressivity by pressing keys in different ways to access more shortcuts, and increased availability by differentiating between normal keyboard input and shortcut input. For example, pressing G with the left index finger and open hand simply enters the letter "G" as would be expected when touch typing (Figure 1a), but when the hand is closed, pressing G with the index finger could mean *Goto Line* (Figure 1b), and when pressed with the right hand, the same G could mean *Goto Page*, *Goto Class*, *Grab*, or *Gap* depending on which finger and what posture is used (Figure 1c). Based on a formative study, we recommend fingers and hand postures to use. A controlled experiment shows that although there is a performance cost compared to simply pressing different keys individually or with a single modifier key, Finger-Aware Shortcuts are comparable to the common and practically-necessary class of shortcuts using multiple modifier keys with the same key.

CHI'16, May 07–12, 2016, San Jose, CA, USA
 ISBN 978-1-4503-3362-7/16/05...$15.00
DOI: http://dx.doi.org/10.1145/2858036.2858355

We accomplish this by monitoring both typing hands using computer vision and the built-in laptop camera augmented with a small reflector. Previous work has tracked hands above a keyboard for mouse-like input or gestures [23, 34, 33, 24, 30], but to our knowledge, identifying which finger has pushed a key for the purpose of issuing shortcut commands has not been explored. Our detection algorithm is packaged into a background service to enable system-wide finger-aware shortcut keys in any application. Based on the guidelines from the formative study, and using our background service, we demonstrate multiple applications illustrating different use cases and mappings for Finger-Aware Shortcuts, including extending the idea to support simultaneous parameter control and two-handed, single key shortcuts.

RELATED WORK

Finger-Aware Shortcuts relate to keyboard shortcut research and methods to augment physical keyboard interaction.

Keyboard Shortcut Availability and Expressivity

Anecdotal evidence suggests some expert users regularly use keyboard shortcuts: Jacob et al. claim video editors do [10], and Knuth says he uses so many Emacs shortcuts that it is "a little bit like playing an organ" [13]. This makes sense considering Lane et al. [16], Karat et al. [11], and others have shown that triggering commands with keys can be faster than pointing at graphical widgets using a mouse. Yet, a comprehensive study by Lane et al. [16] found that keyboard shortcuts are not frequently used by most expert users. McLoone et al. [21] show the uncomfortable and error-prone act of pressing multiple keys simultaneously is a possible factor, Kurtenbach and Buxton [15] and Miller et al. [22] argue this is due to the gulf when transitioning from clicking on graphical menus to pressing shortcut keys, and Grossman et al. [9] and Galitz [6] suggest it is poor mnemonics and visibility.

Regarding the latter, researchers have motivated people with increased shortcut visibility. Grossman et al. [9] provide audio feedback and disable graphical menu items to encourage keyboard shortcut use, Krisler and Alterman [14] display shortcut information and require a shortcut key press before continuing, Malacria et al. [18] and Tak et al. [29] show shortcut information when a modifier key is pressed, and Malacria et al. [19] display an efficiency score to encourage usage.

Our goal is not increased visibility, but shortcut availability and expressivity. Our approach is orthogonal to physical modifications to hardware, such as adding dedicated command keys [21], adding modifier keys to a mouse [26], or a keyboard with actuated, force-sensitive keys [1].

Expanding and Augmenting Keyboard Input

Keyboard input can be expanded by pressing keys in unconventional ways, such as Zhang and Li's [38] method to recognize motion gestures as multiple keys are stroked like a touch screen, but a more common approach is augmenting the simple key switch with additional sensors. The PreSense Keypad [27] senses finger contact before pressing and Dietz et al. [5] developed a pressure sensitive keyboard for continuous input while pressing a key. The Touch-Display Keyboard [3] augments each key with a touch-sensor and a display to increase the input space when the keyboard is used as a single surface and increase shortcut visibility by displaying commands on each key.

Another approach is to augment keyboard input by tracking hand and finger movement above the keyboard. SpaceTop [17] places a transparent screen above the keyboard and a downward facing depth camera to detect finger touch and hand gestures. FlowMouse [34] and TAFFI [33] use a downward facing camera to track hands moving above a conventional keyboard for mouse and touch-screen style input performed mid-air. FingerMouse [23] and AirMouse [24] do the same, but combine mid-air hand movements with key presses (or touch pad taps) to simulate mouse clicks. Taylor et al.'s [30] keyboard with a matrix of infrared proximity sensors embedded between the keys detects a large set of motion gestures that can be easily interlaced with key presses. However, all these projects essentially treat hand gestures and key presses as two independent input techniques.

It is worth noting the body of research combining vision-based mid-air or near-surface hand gesture input with interactive surfaces, but the use case is very different than desktop computing and soft keyboards are not a focus for interaction. For example, Visual Touchpad [20] demonstrates the robustness of their vision-based system by typing on a virtual keyboard, RetroDepth [12] includes an example of a passive retro-reflective keyboard, and a keyboard "SLAP widgets" [32] is a motivating example for the idea of tangible, translucent widgets for multitouch tabletops.

Finger Identification for Physical Buttons

Using finger identification for enhanced multi-touch input is an increasingly active area (see Goguey et al. for a survey [7] and a performance evaluation [8]), but this is tangential to our focus on physical keyboards. More relevant are Sugiura and Koseki [28] and Wang and Canny [31] who both introduce the general idea of pressing a physical button with different fingers to activate different commands. However, both are early proof-of-concept studies using devices that can hardly be called "buttons", much less a full keyboard: Sugiura and Koseki use a commercial fingerprint scanner and Wang and Canny use a piece of glass. Wang and Canny evaluate time to switch fingers, we evaluate the more practical aspect of performance and preferences when using different fingers for shortcuts after typing and pointing tasks. Neither applies this to a physical keyboard, adds the idea of hand postures, or explores the performance and preference design space for shortcut-like command activation in real applications.

FINGER-AWARE SHORTCUTS

We provide details of standard shortcut keys and discuss how Finger-Aware Shortcuts could replace or augment them.

Keyboard Shortcuts

Our focus is on two primary types of shortcuts: single keys (e.g. B for *Brush*) and keys pressed while one or more modifier keys are held down (e.g. ⌘+C for *Copy*). Other more

complex variations exist such as keys pressed after a special command mode is entered (e.g. ctrl+D, then K for *Tasks*), but this is not our focus.

Single "unmodified" shortcuts are practical only when keys are not used for text entry, so this approach is not suited to word processors or text editors. However, they are common in graphical direct manipulation applications like photo editors. A common mapping strategy is to choose keys that correspond to the first letter of the command for a semantic connection. Even with a handful of commands, the first letter key will already be taken leading to more arbitrary mappings (e.g. R for *Blur*). The number of available shortcut commands is also limited by the number of keyboard keys.

A more generalized method is pressing a single key while holding one or more modifier keys. A single modifier can be used, like ⌘+P for *Print*. The ideal mapping uses the first letter of the command, but collisions are unavoidable. Consider ⌘+V for *Paste*. The semantic link could be how "V" resembles an insertion mark, but more likely it was chosen for non-semantic reasons. Since ⌘+C is used for *Copy* and *Paste* is often used immediately after, V suggests a proximity mapping since the two shortcut keys are side-by-side. To expand the input space and enable more meaningful mappings, different modifiers may be used like ⌘+F for *Find* and ctrl+F for *Format*. Or, multiple modifiers can be used together like ⌘+⇧+F for *Find in Project*, ctrl+⌘+F for *Full Screen Mode*, and ctrl+⇧+⌘+F for *Distraction Free Mode*.

Finger-Aware Shortcuts

The central concept is to interpret key presses differently depending on which finger is used and whether the remaining fingers on the same hand are spread open or bent closed. In theory, this provides 20 different ways to interpret each key press (10 fingers × 2 hand postures). In actuality, the formative study in the next section shows 16 are actually reasonable with 8 identified as best. This increased expressivity can be exploited as Finger-Aware Shortcuts.

Text-entry applications can have single "unmodified" shortcut keys based on standard touch typing finger-to-key mappings [2]. When the usual finger presses a key while typing, the alphanumeric character is sent (e.g. pressing G with the left index finger sends "G"). But, when a key is pressed with a finger not normally used while typing, it sends a command (e.g. pressing G with the left middle finger invokes *Goto Line*). This strategy can be relaxed for less-proficient typists by differentiating by open and closed hand postures: typing with an open hand enters text but pressing keys with one finger and a closed hand activates shortcut commands.

Different fingers can map related commands to the same key. For example, pressing C with the right index finger, open hand for *Copy*, pressing C with the same finger, closed hand for *Cut*, and pressing C with the right middle finger for *Paste*. This enables two first letter mappings (*Cut* and *Copy*) and provides a proximity mapping (*Paste*). New kinds of semantic relations can be created based on fingers and postures. For example, using the right thumb, closed hand to issue global commands like *New Document* and the left thumb, closed hand to issue contextual commands like *New Layer*.

A standard modifier key can be combined with finger-specific actions to augment current shortcuts. As an example, accessing a family of commands by holding ⌘ then pressing a key with different fingers, such as F with the left index for *Find*, the left middle for *Find and Replace*, or the left ring for *Find in Project*. Specific fingers could be mapped to common modifier keys used with ⌘. For example, holding ⌘ then pressing a key with the left index finger could mean ⌘+⇧, the left middle ctrl+⌘, or the left ring ctrl+⇧+⌘.

Finger-Aware Shortcuts could potentially benefit expert users using large numbers of commands. However, all of these ideas should be based on performance and preference when using different fingers and hand postures for key presses, which is the goal of the following study.

EXPERIMENT 1: FORMATIVE STUDY

The purpose of this study is to identify a set of postures that maximize performance and minimize error and fatigue. We measured time, error, and subjective preference for different postures in different conditions. The results of this study provided data for training and testing our posture recognition algorithm and enable us to empirically derive guidelines for selecting suitable postures.

Participants and Apparatus

We recruited 21 right-handed participants (9 female, mean age 25.4, SD = 4.9). All reported extensive experience with desktop or laptop computers. Self-reported average weekly computer use ranged from 28 to 84 hours (M = 50.7, SD = 12.6). A one minute typing speed test was performed on participants at the beginning of the experiment. Mean typing speed was 47.9 words-per-minute (SD = 12.7) with a mean error rate of 3.11% (SD = 3.51%).

Note initial analysis found Participant 17 had a very high error rate 11.4% for the third task (explained in Task and Stimuli) compared to the mean error rate 2.47% (SD = 2.61%) across all participants. This participant was removed, and Participant 21 was recruited to keep the design balanced.

The experiment was performed on a MacBook Pro with a 15-inch display and a QWERTY keyboard. The software was written as a web application in Google Chrome hosted from a local server. The built-in camera recorded the keyboard area with the aid of an Osmo reflector (playosmo.com). To simplify background subtraction, a green keyboard cover and green material covered the laptop. A 20-inch square softbox was placed over the laptop at a suitable distance to provide consistent lighting. Hand colours were sampled.

Task and Stimuli

Each trial in our experiment consists of three tasks: a posture task, a reference task, and a repetition of the posture task (Figure 2a,b,c). Upon completion of each task, there was a ding sound with different pitches to indicate success. The accompanying video demonstrates the tasks.

In posture tasks, participants were required to use a certain posture to press a certain key. A posture is defined with 3 independent variables: HAND (LEFTHAND, RIGHTHAND), FORM (OPEN, CLOSED), and FINGER (THUMB, INDEX, MIDDLE, RING, LITTLE). We define 3 KEYAREAS for each finger: keys it normally presses when typing (HOMEAREA), keys at the extreme left (LEFTAREA), and keys at the extreme right (RIGHTAREA) on the keyboard (Figure 2d). HOMEAREA includes the home row key of a finger and the two keys above and under that key. For example, the left middle finger's home row key is D, so its HOMEAREA includes D (home row key), E (above), and C (under). LEFTAREA has Q, A, Z. RIGHTAREA has P, ;, /. The thumb's HOMEAREA is defined as only SPACE. Due to a collision with their home keys, little fingers have W, S, X as LEFTAREA and O, L, . as RIGHTAREA.

In the reference task, participants were required to type three characters or click a button. Reference tasks are denoted with REFTASK (TYPING, POINTING). They were used to simulate real-world scenarios of keyboard interactions. In TYPING, participants were prompted with three characters and asked to sequentially type them (Figure 2b). The characters were randomly selected from the home row: one from left (A,S,D,F), one from right (J,K,L,;), and one from either. The purpose is to force participants to put both hands onto the keyboard. In POINTING, participants were asked to use the touchpad to click a button that appears on the screen. The button could appear either on the left or right, always different from the last appearance. This forces participants to use the touchpad.

Participants were instructed to remember the posture and key in the first task and perform the third task quickly and accurately. No requirement was imposed for the reference task.

Errors and Task Time

A posture task is successful if a participant uses the correct hand, form, and finger to press the correct key. When a wrong key was pressed, participants were immediately informed and would not be able to continue until the correct key was pressed. Posture errors like using a wrong hand, form, or finger were ignored during the experiment and manually classified after the study using images captured by the camera and reflector. A reference task is successful if the correct characters are typed in sequence or the button is clicked. An error occurs when the expected character is not typed or a click does not happen on the button. When any error occurred except posture errors, a low-frequency basso sound was played.

Three durations were recorded for each trial, corresponding to the three tasks. Duration 1 starts when the start button is pressed in the beginning of a block or the last task is completed. It ends when the correct key is pressed. Duration 2 starts immediately after the first task is completed and ends when all three characters are typed or the button is clicked. Duration 3 starts immediately after the second task is completed and ends when the correct key is pressed.

We are interested in the time and error when participants are cognitively prepared for the key to push and the posture to use after a reference task, so the third task was used for analysis. Note the first task allows participants to rehearse the required posture to key, the second task simulates keyboard or touchpad use, and the third task provides the best estimation for time and error measures of practised performance.

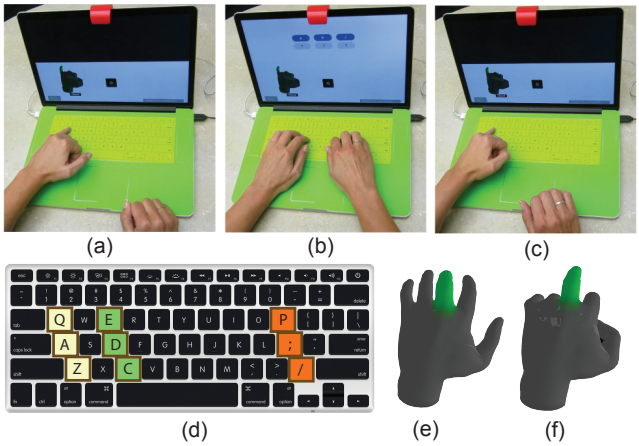

Figure 2: Experiment tasks: (a) prompt for finger and key with initial press; (b) typing reference task; (c) final key press. (d) Key area, example for left middle finger: home area (coloured green in this figure); left area (yellow); right area (red). Hand form examples: (e) left middle, hand open; (f) left middle, hand closed.

Design

The experiment design is within-subject, repeated measures, and full factorial. All trials for each type of FORM were presented together in counterbalanced order. For each FORM, participants performed 3 blocks of measured trials. Each block presented all trials for each combination of HAND, FINGER, KEYAREA, and REFTASK in random order. A short training containing 8 randomly selected trials for each FORM was performed prior to the formal experiment. Short rest breaks were enforced at the end of each block.

In summary: 3 BLOCKS × 2 FORMS × 2 HANDS × 5 FINGERS × 2 REFTASKS × 3 KEYAREAS = 360 trials per participant.

Results

Repeated measures ANOVA and pairwise t-tests with Holm correction were used for all measures[1]. Because measures for preference exhibited non-normality, they were transformed using Aligned Rank Transform [35]. Trials were aggregated by participant and the factors being analysed. Time data were aggregated using the median.

Learning Effect

Overall, BLOCK had a main effect on time ($F_{1.35,25.71} = 30.91$, $p < .0001$, $\eta_p^2 = .619$), but not on error. Post hoc tests found block 1 significantly slower than blocks 2 and 3 (both $p < .0001$) and block 2 significantly slower than block 3 ($p < .005$), suggesting a learning

[1]When the assumption of sphericity was violated, we corrected the degrees of freedom using Greenhouse-Geisser (Greenhouse-Geisser's $\epsilon < 0.75$) or Huynh-Feldt (Greenhouse-Geisser's $\epsilon \geq 0.75$).

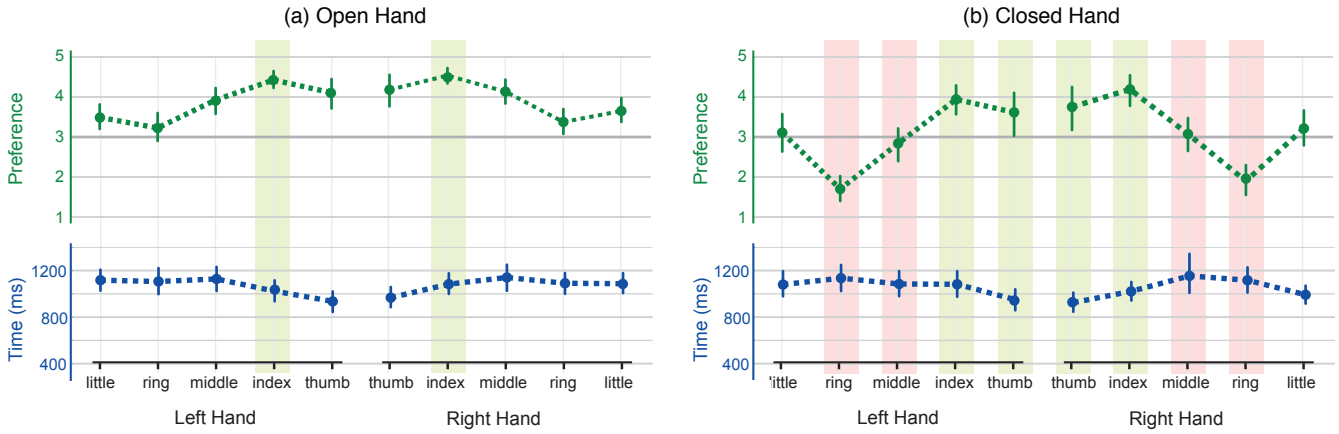

Figure 3: Time and preference by HAND and FINGER for: (a) open hand form; (b) closed hand form. For each FORM, green shading indicates Tier 1 fingers (best), red shading indicates Tier 3 fingers (to avoid). Note FINGER is a categorical variable, dashed lines connecting fingers used for readability only. Error bars are 95% confidence intervals.

effect across all blocks. In all subsequent analysis, we use only block 3 for the best estimation of practised performance.

Error Rate

No main effect was found for REFTASK, FORM, HAND, FINGER, or KEYAREA on error rate and the overall mean error rate was a very good 1.88% (SD = 1.83%). The total error rate breakdown is 1.08% (SD = 1.24%) for pressing the wrong key and 0.79% (SD = 1.06%) for using a wrong posture when the correct key is pressed.

Time

The mean time across all participants was 1060.85ms (SD = 299.52). There was a main effect of REFTASK on time ($F_{1,19} = 59.769$, $p < .0001$, $\eta_p^2 = .759$). Time with POINTING as the reference task was significantly faster than TYPING by 346ms. There was a main effect of FINGER on time ($F_{4,76} = 10.853$, $p < .0001$, $\eta_p^2 = .364$) (Figure 3). Post hoc tests showed THUMB was significantly faster than INDEX and LITTLE (both $p < .05$) and MIDDLE and RING (both $p < .001$). As would be expected, there was a main effect of KEYAREA on time ($F_{2,38} = 23.892$, $p < .0001$, $\eta_p^2 = .557$). Post hoc tests showed pressing keys in HOMEAREA was significantly faster than LEFTAREA by 100ms ($p < .0005$) and RIGHTAREA by 153ms ($p < .0001$), LEFTAREA was significantly faster than RIGHTAREA by 53ms ($p < .05$). No main effect was found for hand or form on time suggesting similar time performance for left and right hands and open and closed forms.

A further examination found an interaction between KEYAREA and HAND on time ($F_{1.39,16.40} = 5.527$, $p < .05$, $\eta_p^2 = .155$). Post hoc tests found LEFTHAND was significantly slower when pressing keys in RIGHTAREA compared to LEFTAREA by 145ms ($p < .05$). No pairwise difference was found for comparisons involving RIGHTHAND or HOMEAREA. Regarding different fingers, there was an interaction between KEYAREA and FINGER on time ($F_{8,152} = 7.132$, $p < .0001$, $\eta_p^2 = .273$). Post hoc tests did not find any difference between pressing LEFTAREA and RIGHTAREA with any finger, but there were differences involving HOMEAREA for all fingers except INDEX. THUMB had the most pronounced difference where HOMEAREA was significantly faster than LEFTAREA by 273ms and RIGHTAREA by 340ms (both $p < .0001$). MIDDLE, RING, and LITTLE had more moderate differences with HOMEAREA significantly faster than RIGHTAREA and/or LEFTAREA by 130ms to 150ms (all $p < .05$).

There was an interaction between REFTASK and FINGER on time ($F_{2.78,52.73} = 5.877$, $p < .005$, $\eta_p^2 = .236$). Post hoc tests showed that for each finger, durations after the POINTING reference task were significantly faster than TYPING ($p < .0005$ for THUMB; $p < .0001$ for others). There was no interaction between REFTASK and HAND, HAND and FINGER, or FORM and FINGER on time. An examination of the data also indicates a symmetric pattern for fingers across hands for these factors (Figure 3).

Preference

At the end of the experiment, participants were asked to consider accuracy, speed, and fatigue and provide a numerical preference score for each combination of HAND, FINGER, and FORM. Preference scores used a real numbered, continuous scale from 1 (least preferred) to 5 (most preferred). Decimal ratings such as 3.5 were permitted.

There was a main effect of FINGER on preference ($F_{2.39,45.47} = 21.962$, $p < .0001$, $\eta_p^2 = .536$). Post hoc tests found RING (2.55) was less preferred than all other fingers (all $p < .005$). Differences between other fingers suggest a partial ranking with INDEX (4.27) preferred over MIDDLE (3.49) and LITTLE (3.37) (both $p < .0005$), and THUMB (3.91) preferred over LITTLE ($p < .05$). There was a main effect of FORM on preference ($F_{1,19} = 33.675$, $p < .0001$, $\eta_p^2 = .639$). Hand form OPEN (3.90) was preferred over CLOSED (3.13). There was also a main effect of HAND on preference ($F_{1,19} = 18.166$, $p < .0005$, $\eta_p^2 = .489$). RIGHTHAND (3.60) was preferred over LEFTHAND (3.44). This may be attributed to all participants being right-handed.

There was a large interaction of FINGER and FORM on preference ($F_{4,76} = 17.249$, $p < .0001$, $\eta_p^2 = .476$). Most interesting is that post hoc tests found RING OPEN (3.3) was more preferred than RING CLOSED (1.8) ($p < .01$). No interaction was found between FINGER and HAND, FORM and HAND, or FINGER, FORM, and HAND on preference.

In the interview, 3 participants indicated their dislike of THUMB OPEN, explaining the hand could hit the screen when pressing the upper keys; 2 participants expressed their aversion to MIDDLE CLOSED due to bad social implications.

Discussion

Our study results show that Finger-Aware Shortcuts, performed with any finger with either open or closed hand forms, are acceptable in terms of speed. A consistent learning effect and low error rate suggests the technique is easy to learn and perform. Pressing left and right areas of the keyboard have comparable time for all fingers, regardless of their home areas. As expected, pressing keys in the home area is significantly faster than left or right area, but the difference is reasonably small. The significant difference between reference tasks suggests that Finger-Aware Shortcuts will be faster in pointing-intense applications compared to typing-intense applications. However, both times are within a reasonable and practical range. Overall, this study shows that pressing a single key in 16 different ways is feasible, supporting our goal of increasing keyboard expressivity and availability.

Guideline for Selecting Postures

We use time and preference results to provide a three-tier guideline for selecting postures. Tier 1 are most recommended postures and Tier 3 are least recommended (see green and red shading in Figure 3). Since time and preference are symmetric between hands, we simplify our discussion to 5 fingers × 2 forms regardless of hand. The index finger has consistently high performance and preference for open and closed forms, so it is Tier 1. The thumb is comparable, but people with larger hands hit the screen when using the open form, so we place thumb closed in Tier 1 and thumb open in Tier 2. Although the performance of middle, ring, and little fingers with an open hand are good, they are less preferred than the index and thumb. We place them in Tier 2. When the hand is closed, the middle finger has similar measures compared to the little, but this posture has negative social implications, so it is Tier 3. The ring finger with closed form is least preferred, so it is also Tier 3.

In summary, Tier 1 postures are index (hand open/closed) and thumb (hand closed). Tier 2 postures are thumb (hand open), middle (hand open), ring (hand open), little (hand open/closed). Tier 3 postures are middle (hand closed) and ring (hand closed). In practice, Tier 1 postures should be mapped with most frequently used commands; Tier 2 postures should be mapped with less frequent, secondary functions; and Tier 3 postures should be avoided.

IMPLEMENTATION

In this section, we describe an algorithm that identifies the hand and finger on the keyboard when a key is pressed. The algorithm recognizes all postures that are in Tier 1 and 2 of our guideline. We used the same apparatus as Experiment 1, including the webcam reflector (Figure 4a) and green material covering the keyboard face that facilitates background subtraction (Figure 4b). Our simple prototype can be easily reproduced and might be further improved with dedicated sensors and improved background models.

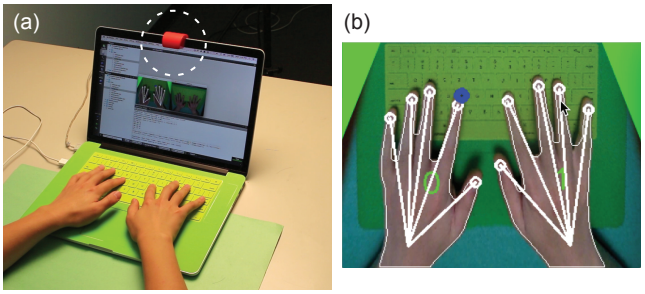

Figure 4: Apparatus: (a) A reflector directs the webcam light path to the keyboard; (b) aided by a green keyboard cover and laptop skin, the hands are isolated and tracked in a rectified frame.

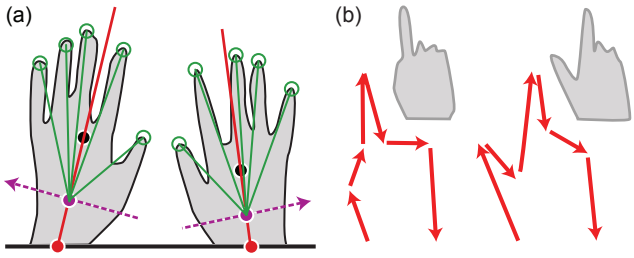

Figure 5: Algorithm: (a) hand feature extraction showing wrist reference points as red dots, palm centre as purple dot, principle axis as red line, and perpendicular axis as dashed purple line; (b) stroke recognition approach for hand and finger identification.

Keyboard Localization

To reliably determine which finger is pressing a key, we use a keyboard cover with a slightly different shade than the green laptop skin. The cover's colour was sampled in advance. We use this information to extract the keyboard area and automatically localize the four corners of the keyboard by approximating a quadrilateral. The perspective of the image frame is corrected by rectifying the quadrilateral to a rectangle. Key positions are mapped from the physical keyboard to coordinates in this rectangle.

Hand Feature Extraction

Hand skin and background samples from Experiment 1 were converted to the HSV colour space to train a Gaussian naive Bayes classifier. In detection, the classifier flags the pixels as skin, but the pixels are often disconnected or contain false positives. Performing a morphological close followed by an open reduces noise and brings each hand contour together. The contours are then smoothed by sequentially applying blurring, dilating, thresholding, and eroding. To eliminate dark areas between two fingers, Sobel operators are applied to the contour areas to identify sharp changes in derivatives of brightness. The contours are smoothed one more time.

Two points and two axes are calculated from each hand contour to facilitate identifying fingers and hand shape. With the assumption that computer users always interact with the keyboard by reaching their hands forward, we calculate a *wrist reference* by simply taking the intersection midpoint between the forearm and the bottom of the camera's view (Figure 5a). We also estimate the *palm centre* by calculating a weighted average of the wrist reference and centroid of the contour. A *principal axis* is formed with the vector from the wrist reference to the palm centre. This provides an estimation of the orientation of the hand. A *perpendicular axis* is perpendicular to the principal axis at the palm centre pointing outwards. This axis is used to reduce false positives during fingertip detection and provide a reference for sorting vectors of the wrist reference and fingertips.

Fingertip Localization

A similar approach to Yoruk et al. [37] is used for localizing fingertips. We compute distances from the wrist reference to each point on the contour in sequence, then local maxima are candidates for fingertips. False positives on the wrist reference side of the perpendicular axis are eliminated. Remaining candidates are mostly extended fingers. Bent fingers can be falsely detected due to the slight curvature at the knuckle, but we minimize this by tuning the curvature threshold.

Hand and Finger Identification

Using the extracted hand features and localized fingertips, we applied two approaches for hand identification: stroke recognition and heuristic rules. Both distinguish between left and right hands and open or closed forms. We used stroke recognition in Experiment 2 and heuristic rules in Demonstrations.

Stroke Recognition Approach — A hand contour can be normalized as a single "stroke" from the bottom left corner to the bottom right corner, ignoring the intersection between the forearm and the bottom of the frame. Therefore, techniques used for processing and recognizing strokes can be applied to classify hand shapes (Figure 5b). We use 1€ Filter [4] to smooth and $1 Recognizer [36] to recognize the stroke. The 1€ Filter is a first-order low-pass filter for reducing noisy signals. Applying it produces a more normalized stroke which aids the $1 Recognizer algorithm. The $1 Recognizer is a 2D single stroke recognizer. Each posture, hand side, and form is labelled with a code (such as "R01000" if right hand with the index finger extended and other fingers closed). We train the $1 Recognizer with the filtered strokes and corresponding labels. When a single finger is extended, the finger used for pressing the key is immediately identified.

Heuristic Rule Approach — As a simple and practical alternative, we use heuristics to identify hands. Hand side is jointly determined with the relative positions of the wrist reference and the direction of the principal axis. The frame is equally divided into three parts horizontally. For example, if the wrist reference falls into the left part, it is a left hand. If the wrist reference falls into the middle part, the opposite side of the principal axis direction is the hand side. With simple heuristics, hand form is determined using the distance of each fingertip to the palm centre and their vertical differences. When the maximum of the distances is greater than all other distance by a threshold, either index or little is extended. When the difference of the maximum and minimum of the vertical differences is smaller than a threshold, the thumb is extended. In all other conditions, we consider the hand open.

If an open hand is recognized by either approach, fingertips are sorted in descending order based on the angle between the perpendicular axis and the vector from the palm centre to each fingertip. A larger angle indicates the finger is closer to the thumb. We compare the distance from each finger to the key and identify the finger using its order.

Hand and Finger Tracking

We use a Kalman Filter to track each identified fingertip. The filter models fingertip movement as uniform motion. When a new observation is not presented, a fingertip position is predicted. We use dynamic programming to match new fingertip positions with predicted positions. Cartesian distances are employed to measure difference between two positions. The dynamic programming reduces the computational cost by assuming the mapping from one sorted fingertip to another is monotonic. It also yields a best matching score by calculating the minimum distance sum of two hands. With the best matching scores, we can also trivially track hands.

System-Wide Background Service

We built our algorithm as a background service in Apple OS X. The service has two parts: an OpenCV posture recognizer and a Cocoa key event interceptor. The interceptor traps key events from the operating system using Quartz Event Services. When a key down is received, it sends the key code and timestamp to an intermediate in-memory event queue. The recognizer receives the event and searches in a 3s cache for an unprocessed frame with the closest timestamp. The posture is then recognized and the interceptor is notified via another queue. The interceptor rewrites the event according to a configurable mapping between Finger-Aware Shortcuts and equivalent shortcut keys. The event is fired immediately after receiving the posture or releasing the key. Processing a shortcut using our background service takes approximately 77ms, empirically shorter than the duration of a key press and hardly perceivable. With accessibility features in OS X, we can also detect which application is active and interpret key events as application-specific commands.

EXPERIMENT 2: COMPARISON WITH SHORTCUT KEYS

The goal of this study is to compare Finger-Aware Shortcuts with current keyboard shortcut approaches. Time and error are used as primary performance metrics, but we also examine cognitive differences using perceived workload and frequency of viewing the command mappings. The general format of the experiment is similar to Experiment 1.

Participants and Apparatus

A subset of 8 participants (3 female) from Experiment 1 were recruited, aged from 23 to 37 ($M = 25.6$, $SD = 4.7$). The same set of apparatus was employed. The stroke-based recognition algorithm (described above) was used in real-time.

Task and Stimuli

In each trial, participants completed a reference task followed by a shortcut task. The reference task simulates real-world scenarios of keyboard interactions with the same TYPING and POINTING reference tasks from Experiment 1.

Our main focus is the shortcut task where participants issue one of 6 "commands" using a shortcut technique: either Finger-Aware Shortcuts or a standard keyboard shortcut method. The commands were represented as two sets of three four-letter English words. All words in each set begin with the same letter (e.g. W: *Wack, Wool, Whim*, P: *Peek, Pill, Pump*).

We tested for 3 variations representing how commands are typically mapped to standard keyboard-only shortcuts:

3KEY*: 3 Keys Without Modifier* — The set of commands are mapped to different keys without any modifier key. Only one key can correspond to the first letter of the command, other keys are chosen arbitrarily. Using the 'W' example set, this could create the following mapping: W for *Wack*; S for *Wool*; and R for *Whim*. This variation is based on a common method for mapping shortcut keys in drawing applications where text entry is modal, for example in Adobe Illustrator, S means *Scale*, V means *Select*, and K means *Slice*. We expect 3KEY will be fast since it is simply pressing a key, but there may be cognitive overhead from the conflicting key and first word letter.

3KEY1MOD*: 3 Keys using 1 Modifier* — The set of commands are mapped to different keys using a single modifier key. As above, only one key can correspond to the first letter of the command. We use the ⌘ (command) modifier. Using the 'W' example set, this could create the following mapping: ⌘+W for *Wack*; ⌘+D for *Wool*; and ⌘+A for *Whim*. This variation is based on an analogous common shortcut key mapping: ⌘+C for *Copy*; ⌘+X for *Cut*; and ⌘+W for *Close*. We expect 3KEY1MOD to be slower than 3KEY due to the extra modifier press, but the same conflict between word and key letter could introduce cognitive overhead.

1KEY3MOD*: 1 Key using 3 Modifiers* — All commands in a set are mapped to the same key using three different modifier key combinations. The key corresponds to the common first letter in the command set, for example *Wack, Wool, Whim* are all mapped to W. Three modifier key combinations were used: ⌘ (command), ⌘+⇧ (command + shift), and ⌥ (alternate). For the example set, this creates the following mapping: ⌘+W for *Wack*; ⌘+⇧+W for *Wool*; and ⌥+W for *Whim*. This variation is based on applications that use different modifier keys to map related commands to the same key, such as ⌘+S for *Save*; ⌘+⇧+S for *Save As ...*; and ⌥+S for *Save Copy*. We expect 1KEY3MOD to be slower than 3KEY, but it may have less cognitive overhead without a conflict between word and key letter.

We compare these to Finger-Aware Shortcuts:

FINGERAWARE*: 1 Key with 3 Finger Aware Postures* — All commands in a set are mapped to the same key using three different postures. The key corresponds to the common first letter in the command set, for example *Wack, Wool, Whim* are all mapped to W. Three postures were used: INDEX OPEN, INDEX CLOSED, and MIDDLE OPEN. All postures were performed with the right hand. These postures were selected according to the guideline derived from the formative study: two postures are from Tier 1 (INDEX OPEN, INDEX CLOSED) and one Tier 2 (MIDDLE OPEN). Like 1KEY3MOD, there is no conflict between word and key letter. However, we expect it to be slower than 3KEY given overhead for using a specific finger and posture. The question is whether it is comparable to 1KEY3MOD or 3KEY1MOD since it could be an alternative to these common techniques.

Common Task Details

With every SHORTCUT, the two sets of 3 commands are mapped to a subset of 12 keys: 6 keys on the left (W, E, R, S, D, F) and 6 on the right (P, O, I, L, K, J). The two command sets were chosen such that the first letter of one set matched one of the left keys and the other matched one of the right keys. Parts of the keyboard are denoted with KEYPART (LEFTPART, RIGHTPART).

For single key mappings (1KEY3MOD, FINGERAWARE), all commands in a set were assigned to the key matching the first letter. For multiple key mappings (3KEY, 3KEY1MOD), one command from each set was assigned to the key matching the first letter. The remaining key mappings were randomly selected without duplicates from the remaining 10 keys.

All shortcut mappings were displayed in a cue card, rendered as two columns of 3 commands. The cue card could be viewed any time by pressing SPACE, but participants were encouraged to memorize the mappings. The current trial was restarted if the cue card was viewed. We logged the number of view (CC Count) and viewing duration (CC Duration) as additional factors to test learning effects.

Errors and Task Time

We are only concerned with errors during shortcut tasks. An error occurs when the wrong shortcut keys are used for the prompted command word. Participants immediately repeated the same trial beginning from the reference task for a maximum of 3 tries. This repeated trial design maximizes error-free trials for analysis. With FINGERAWARE, errors may also be caused by incorrect posture recognition. These errors appeared as trial errors to participants. but they were informed that FINGERAWARE could have false negatives prior to the experiment. We manually corrected for these false errors (accounting for 10.9% overall) before analysis. Most were due to capture quality issues such as fingers cropped by capture area, shadows, and motion blur, which could be addressed with a wider angle, faster, and more sensitive camera. Others were caused by one hand occluding the other, which could be addressed with a depth sensor.

Design

The experiment design is within-subject, repeated measures, and full factorial. All trials for each variation of SHORTCUT were presented together, with their order counterbalanced using a Latin Square. For each SHORTCUT, participants performed 5 blocks of measured trials. Each block presented all

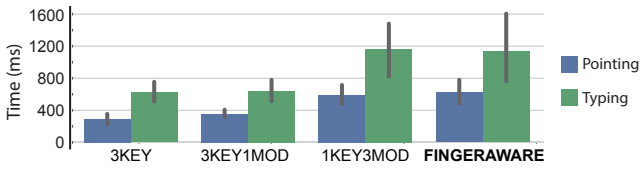

Figure 6: Time by SHORTCUT technique by reference task.

trials for all combinations of COMMANDS and REFTASKS in random order for 3 REPETITIONS.

In summary: 4 SHORTCUTS × 5 BLOCKS × 6 COMMANDS × 2 REFTASKS × 3 REPETITIONS = 720 trials per participant.

Results

Analysis used the same approach as Experiment 1. CC Count and CC Duration were aggregated using sum. All subjective data were transformed with Aligned Rank Transform.

Learning Effect

BLOCK had a main effect on time ($F_{4,28} = 14.756$, $p < .0001$, $\eta_p^2 = .678$). Post hoc tests found block 1 significantly slower than blocks 4 and 5, and block 2 significantly slower than block 4 (all $p < .05$). BLOCK had a main effect on error ($F_{4,28} = 5.298$, $p < .005$, $\eta_p^2 = .431$), but no post hoc differences. BLOCK had a main effect on CC Duration ($F_{1.21,8.46} = 54.196$, $p < .0001$, $\eta_p^2 = .886$). Post hoc tests show block 1 had a longer duration than all other blocks (all $p < .005$). Given these learning effects, only blocks 4 and 5 are used in subsequent analysis.

Error Rate

Excluding false negatives in FINGERAWARE, the mean error rate across all participants is 8.7% with SD = 5.53% (3KEY 8.9%, 3KEY1MOD 9.0%, 1KEY3MOD 9.9%, FINGERAWARE 6.9%). There was no main effect of SHORTCUT or REFTASK on error, nor was there a main effect of SHORTCUT on error for REFTASK. Analysing each SHORTCUT separately found a main effect of REFTASK on error in 3KEY ($F_{1,7} = 40.111$, $p < .0005$, $\eta_p^2 = .851$): TYPING (M = 12.15%, SD = 5.34%) had a significantly higher error rate than POINTING (M = 5.56%, SD = 6.47%).

Time

The mean time across all participants was 629ms (SD = 190). Overall, there was a main effect of SHORTCUT on time ($F_{1.54,10.81} = 9.809$, $p < .0005$, $\eta_p^2 = .584$). Post hoc tests showed 3KEY faster than 1KEY3MOD and FINGERAWARE both by 421ms ($p < .005$), and 3KEY1MOD faster than 1KEY3MOD and FINGERAWARE, both by 377ms ($p < .05$).

Analysing each REFTASK separately (Figure 6), SHORTCUT had a main effect on time in POINTING ($F_{3,21} = 10.292$, $p < .0005$, $\eta_p^2 = .595$) and in TYPING ($F_{1.58,11.05} = 7.676$, $p < .05$, $\eta_p^2 = .523$). Post hoc tests revealed that in POINTING, 3KEY was significantly faster than 1KEY3MOD by 303ms and FINGERAWARE by 329ms; 3KEY1MOD was significantly faster than 1KEY3MOD by 238ms and FINGERAWARE by 264ms (all $p < .05$). In TYPING, 1KEY3MOD was significantly slower than 3KEY by 539ms and 3KEY1MOD by 515ms (both $p < .05$).

Perceived Workload

At the end of the experiment, sub-scales of NASA-TLX were used to elicit participants' perceived workload. Mental demand, physical demand, performance, effort, and frustration for each SHORTCUT were: 3KEY 8.1, 4.4, 8.6, 7.3, and 6.6; 3KEY1MOD 12.6, 10.3, 9.8, 10.9, and 10.8; 1KEY3MOD 13.9, 13.1, 11.4, 13.4, and 12.9; FINGERAWARE 14.0, 12.7, 13.0, 12.7, and 13.9, respectively.

SHORTCUT had a main effect on mental demand ($F_{3,21} = 3.771$, $p < .05$, $\eta_p^2 = .350$), with post hoc tests showing 3KEY had lower demand than 1KEY3MOD and FINGERAWARE (both $p < .05$). SHORTCUT had a main effect on physical demand ($F_{3,21} = 10.595$, $p < .0005$, $\eta_p^2 = .602$), with post hoc tests showing 3KEY had lower demand compared to all others (all $p < .05$). SHORTCUT had a main effect on effort ($F_{3,21} = 6.785$, $p < .005$, $\eta_p^2 = .492$), with post hoc tests showing 3KEY had lower effort than 1KEY3MOD ($p < .005$) and FINGERAWARE ($p < .01$). SHORTCUT had a main effect on frustration ($F_{3,21} = 4.152$, $p < .05$, $\eta_p^2 = .372$), with post hoc tests showing 3KEY had lower frustration than FINGERAWARE ($p < .05$). There was no main effect of SHORTCUT on performance.

Memorisation

A real numbered, continuous scale was provided for each SHORTCUT, describing the ease of memorising the command-shortcut mappings. The scale is from 1 (hardest to remember) to 5 (easiest to remember). The results were 3KEY 4.0, 3KEY1MOD 2.2, 1KEY3MOD 2.8, and FINGERAWARE 3.0. There was a main effect of SHORTCUT on ease of memorisation ($F_{3,21} = 3.423$, $p < .05$, $\eta_p^2 = .328$), with post hoc tests showing 3KEY easier to remember than 3KEY1MOD ($p < .05$).

Participants were asked about their memorisation strategy for each SHORTCUT. Their descriptions fall into three categories: *a)* memorizing the order of shortcuts and commands independently, then using orders to establish a mapping; *b)* using mnemonics; and *c)* using no special technique. In 3KEY, 6 participants used *b*, 1 used *a*, and 1 used *c*. In 3KEY1MOD, 4 used *b*, 3 used *c*, and 1 used *a*. 1KEY3MOD and FINGERAWARE had the same result: 6 used only *a*, one only used *b*, and 1 used both *a* and *b*.

Discussion

Our finding that single key shortcuts are fast was expected – this is simply pressing a key. Despite the conflicts between command letters and key letters, such strong performance when mapping single keys with or without one modifier is surprising. We expected a cognitive processing overhead, but the result suggests this was less of a concern. This may be because command names with key letters are easily pronounced compared to modifier keys or postures, so people can use a phonological loop to form short-term memory. However, this benefit may be due to the relatively small number of commands and short training time. It is possible that increasing the number of commands or performing a longer study could

reveal different patterns. Regardless, in real world usage, single key shortcuts are limited to non-text modes and both single key techniques are limited in the number of available keys.

It is encouraging that Finger-Aware Shortcuts achieved similar performance with one key with multiple modifiers. As reflected by the memorisation patterns used by participants, the two techniques are analogous in that different postures act like different modifier keys. It is important to recognize that using specific fingers and postures to press keys is not currently a well practised task. With more time, advantages for motor memory are likely to develop, further decreasing performance time. In addition, participants explained that higher frustration scores with Finger-Aware Shortcuts were due to occasional false detections in the computer vision system. This likely negatively impacted performance. Since participants are already familiar with multiple modifier shortcuts, the comparable performance of Finger-Aware Shortcuts is notable considering the short exposure and "research-quality" posture recognition.

Overall, our study confirmed that Finger-Aware Shortcuts have reasonable performance. With further improvements to recognition robustness and more practice, our technique could be a viable alternative to multiple modifier keys by substituting modifiers with postures. It could also enable single shortcut keys in text-focused applications by simply using the thumb to press keys while typing. It could even be combined with a single modifier key to double the number of single key mappings.

APPLICATION DEMONSTRATIONS

We demonstrate Finger-Aware Shortcuts for complementary commands, up/down controls, and alternative modifiers in *Adobe Photoshop* and *Google Docs*. We also extend for parameter control during a shortcut press and two-handed shortcuts with a single key. The accompanying video also demonstrates these applications.

Complementary Commands

Finger-Aware Shortcuts enable the same key to overload a family of related commands. In Google Docs, pressing H with the left index finger, open hand to change the style of the current line to Heading 1, the middle to Heading 2, and the ring to Heading 3. Text is set to normal with the left thumb, closed hand. The right thumb, closed hand could function like ⌘: pressing C triggers *Copy* and V triggers *Paste*. A slight modification on these commands could be using the left thumb, closed hand to *Paste and Match Style*.

Up/Down Controls

By mapping two fingers to up and down, then pressing related keys, numerical values of different properties can be adjusted. In Photoshop, pressing B with the right index, open hand decreases the brightness of an image, while the right middle increases the brightness. Similarly, pressing C with different fingers adjusts contrast and pressing O adjusts opacity. Up/down controls could be mapped to a non-dominant hand when pointing is intense. When drawing with the brush tool in Photoshop using the right hand, pressing H with the left index or middle finger using an open hand decreases or increases the hue of the brush colour.

Alternate Modifiers

Another way is to map keyboard shortcut modifier combinations to different postures. For example, mapping the ⌘ modifier to the right thumb, closed hand. All existing keyboard shortcuts using only ⌘ can then be triggered by pressing the same key with this thumb posture. In Photoshop, this could *Open File* by pressing O, *Save* by pressing S, etc. Similarly, ⌘+⇧ could be mapped to the right index and closed hand, so pressing N with that posture invokes *Create a New Layer*. In Google Doc, *Bold*, *Italic*, and *Underline* can be triggered in the same way using the right thumb.

Simultaneous Parameter Control

It is possible to track finger movement to control parameters during a key press. For example, pressing and holding a key with the index finger with extended thumb, then moving the thumb to adjust a continuous parameter. For example, adjusting an RGB colour by pressing and holding R to adjust red, G to adjust green, or B to adjust the blue. Thumb movement can be discretized into menu options for variations of a command. For example, holding P with the index finger displays a partial pie menu with items selected with the thumb, like *Print to Printer* and *Print to PDF*.

Two-Handed Key Presses

Special or uncommon commands could even be triggered with two-handed key presses. For example, using both index fingers to simultaneously press L to *Log Out*.

CONCLUSION AND FUTURE WORK

Our work introduces the idea of Finger-Aware Shortcuts and shows it is a viable method for augmenting current shortcut keys, and perhaps used as an alternative to shortcut keys requiring multiple modifiers. The recognition algorithm is robust enough for conducting an experiment and demonstrating the technique, but infallible hand tracking in all environmental conditions remains a barrier for full deployment. If we use an alternate capture set up with the assumption that it could be built into future laptops, hand tracking could be improved further. For example, pairing stereo cameras with infrared illumination and a laptop and keyboard with a retro reflective coating [12] would dramatically increase the reliability of hand and finger extraction. In addition to technical improvements, our findings regarding shortcut mappings and cognitive load require more targeted studies to control and fully understand. A direct comparison between mappings based on fingers and keys under many different variations of command names, keys, and cue card access would be an informative next step. We also believe finger-specific keyboard interaction may have other useful applications. For example, we are exploring how to use the tracking information to teach poor typists correct touch typing fingering.

REFERENCES

1. Gilles Bailly, Thomas Pietrzak, Jonathan Deber, and Daniel J. Wigdor. 2013. Métamorphe: Augmenting Hotkey Usage with Actuated Keys. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. 563–572.
2. Ruth Ben'ary. 1989. *Touch Typing in Ten Lessons* (revised edition ed.). Perigee Books, New York, NY, USA.
3. Florian Block, Hans Gellersen, and Nicolas Villar. 2010. Touch-Display Keyboards: Transforming Keyboards into Interactive Surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. 1145–1154.
4. Géry Casiez, Nicolas Roussel, and Daniel Vogel. 2012. 1 € Filter: A Simple Speed-based Low-pass Filter for Noisy Input in Interactive Systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. 2527–2530.
5. Paul H. Dietz, Benjamin Eidelson, Jonathan Westhues, and Steven Bathiche. 2009. A Practical Pressure Sensitive Computer Keyboard. In *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology (UIST '09)*. 55–58.
6. Wilbert O. Galitz. 2007. *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques* (3, illustrated ed.). John Wiley & Sons, New York, NY, USA.
7. Alix Goguey, Géry Casiez, Thomas Pietrzak, Daniel Vogel, and Nicolas Roussel. 2014. Adoiraccourcix: Multi-touch Command Selection Using Finger Identification. In *Proceedings of the 26th Conference on L'Interaction Homme-Machine (IHM '14)*. 28–37.
8. Alix Goguey, Mathieu Nancel, Géry Casiez, and Daniel Vogel. 2016. The Performance and Preference of Different Fingers and Chords for Pointing, Dragging, and Object Transformation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '16)*. 12p.
9. Tovi Grossman, Pierre Dragicevic, and Ravin Balakrishnan. 2007. Strategies for Accelerating On-line Learning of Hotkeys. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*. 1591–1600.
10. Robert J. K. Jacob, Audrey Girouard, Leanne M. Hirshfield, Michael S. Horn, Orit Shaer, Erin Treacy Solovey, and Jamie Zigelbaum. 2008. Reality-Based Interaction: A Framework for Post-WIMP Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. 201–210.
11. John Karat, James E. McDonald, and Matt Anderson. 1986. A Comparison of Menu Selection Techniques: Touch Panel, Mouse and Keyboard. *International Journal of Man-Machine Studies* 25, 1 (July 1986), 73–88.
12. David Kim, Shahram Izadi, Jakub Dostal, Christoph Rhemann, Cem Keskin, Christopher Zach, Jamie Shotton, Timothy Large, Steven Bathiche, Matthias Niessner, D. Alex Butler, Sean Fanello, and Vivek Pradeep. 2014. RetroDepth: 3D Silhouette Sensing for High-precision Input on and Above Physical Surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. 1377–1386.
13. Donald E. Knuth and Andrew Binstock. 2008. Interview with Donald Knuth. (25 Apr 2008). Retrieved Sep 20, 2015 from `http://www.informit.com/articles/article.aspx?p=1193856`.
14. Brian Krisler and Richard Alterman. 2008. Training Towards Mastery: Overcoming the Active User Paradox. In *Proceedings of the 5th Nordic Conference on Human-computer Interaction: Building Bridges (NordiCHI '08)*. 239–248.
15. Gordon Kurtenbach and William Buxton. 1994. User Learning and Performance with Marking Menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '94)*. 258–264.
16. David M. Lane, H. Albert Napier, S. Camille Peres, and Anikó Sándor. 2005. Hidden Costs of Graphical User Interfaces: Failure to Make the Transition from Menus and Icon Toolbars to Keyboard Shortcuts. *International Journal of Human-Computer Interaction* 18, 2 (May 2005), 133–144.
17. Jinha Lee, Alex Olwal, Hiroshi Ishii, and Cati Boulanger. 2013. SpaceTop: Integrating 2D and Spatial 3D Interactions in a See-through Desktop Environment. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. 189–192.
18. Sylvain Malacria, Gilles Bailly, Joel Harrison, Andy Cockburn, and Carl Gutwin. 2013a. Promoting Hotkey Use through Rehearsal with ExposeHK. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. 573–582.
19. Sylvain Malacria, Joey Scarr, Andy Cockburn, Carl Gutwin, and Tovi Grossman. 2013b. Skillometers: Reflective Widgets That Motivate and Help Users to Improve Performance. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST '13)*. 321–330.
20. Shahzad Malik and Joe Laszlo. 2004. Visual Touchpad: A Two-handed Gestural Input Device. In *Proceedings of the 6th International Conference on Multimodal Interfaces (ICMI '04)*. 289–296.
21. Hugh McLoone, Ken Hinckley, and Edward Cutrell. 2003. Bimanual Interaction on the Microsoft Office Keyboard. In *Proceedings of IFIP TC 13 International Conference on Human-Computer Interaction (INTERACT '03)*. 49–56.

22. Craig S. Miller, Svetlin Denkov, and Richard C. Omanson. 2011. Categorization Costs for Hierarchical Keyboard Commands. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. 2765–2768.
23. Thomas A. Mysliwiec. 1994. FingerMouse: A Freehand Computer Pointing Interface. In *Proceedings of IEEE International Conference on Automatic Face and Gesture Recognition (FG '95)*. 372–377.
24. Michael Ortega and Laurence Nigay. 2009. AirMouse: Finger Gesture for 2D and 3D Interaction. In *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction (INTERACT '09)*. 214–227.
25. S. Camille Peres, Michael D. Fleetwood, Minmin Yang, Franklin P. Tamborello, and Danielle Paige Smith. 2005. Pros, Cons, and Changing Behavior: An Application in the Use of the Keyboard to Issue Commands. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting (HFES '05)*, Vol. 49. 637–641.
26. Thomas Pietrzak, Sylvain Malacria, and Gilles Bailly. 2014. CtrlMouse et TouchCtrl: Duplicating Mode Delimiters on the Mouse. In *Proceedings of the 26th Conference on L'Interaction Homme-Machine (IHM '14)*. 38–47.
27. Jun Rekimoto, Takaaki Ishizawa, Carsten Schwesig, and Haruo Oba. 2003. PreSense: Interaction Techniques for Finger Sensing Input Devices. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology (UIST '03)*. 203–212.
28. Atsushi Sugiura and Yoshiyuki Koseki. 1998. A User Interface Using Fingerprint Recognition: Holding Commands and Data Objects on Fingers. In *Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology (UIST '98)*. 71–79.
29. Susanne Tak, Piet Westendorp, and Iris van Rooij. 2013. Satisficing and the Use of Keyboard Shortcuts: Being Good Enough Is Enough? *Interacting with Computers* 25, 5 (Sept. 2013), 404–416.
30. Stuart Taylor, Cem Keskin, Otmar Hilliges, Shahram Izadi, and John Helmes. 2014. Type-Hover-Swipe in 96 Bytes: A Motion Sensing Mechanical Keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. 1695–1704.
31. Jingtao Wang and John Canny. 2004. FingerSense: Augmenting Expressiveness to Physical Pushing Button by Fingertip Identification. In *CHI '04 Extended Abstracts on Human Factors in Computing Systems (CHI EA '04)*. 1267–1270.
32. Malte Weiss, Julie Wagner, Yvonne Jansen, Roger Jennings, Ramsin Khoshabeh, James D. Hollan, and Jan Borchers. 2009. SLAP Widgets: Bridging the Gap Between Virtual and Physical Controls on Tabletops. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. 481–490.
33. Andrew D. Wilson. 2006. Robust Computer Vision-based Detection of Pinching for One and Two-handed Gesture Input. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology (UIST '06)*. 255–258.
34. Andrew D. Wilson and Edward Cutrell. 2005. FlowMouse: A Computer Vision-Based Pointing and Gesture Input Device. In *Proceedings of IFIP TC 13 International Conference on Human-Computer Interaction (INTERACT '05)*. 565–578.
35. Jacob O. Wobbrock, Leah Findlater, Darren Gergle, and James J. Higgins. 2011. The Aligned Rank Transform for Nonparametric Factorial Analyses Using Only Anova Procedures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. 143–146.
36. Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. 2007. Gestures without Libraries, Toolkits or Training: A $1 Recognizer for User Interface Prototypes. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology (UIST '07)*. 159–168.
37. Erdem Yoruk, Ender Konukoglu, Bulent Sankur, and Jérôme Darbon. 2006. Shape-based Hand Recognition. *Image Processing, IEEE Transactions on* 15, 7 (July 2006), 1803–1815.
38. Haimo Zhang and Yang Li. 2014. GestKeyboard: Enabling Gesture-based Interaction on Ordinary Physical Keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. 1675–1684.