GestureCalc: An Eyes-Free Calculator for Touch Screens

Bindita Chaudhuri[1]*, Leah Perlmutter[1]*, Justin Petelka[2], Philip Garrison[1]
James Fogarty[1], Jacob O. Wobbrock[2], Richard E. Ladner[1]
[1]Paul G. Allen School of Computer Science & Engineering, [2]The Information School
DUB Group | University of Washington, Seattle, WA 98195, USA
[1]{bindita, lrperlmu, philipmg, jfogarty, ladner}@cs.washington.edu, [2]{jpetelka, wobbrock}@uw.edu

ABSTRACT
A digital calculator is one of the most frequently used touch screen applications. However, keypad-based character input in existing calculator applications requires precise, targeted key presses that are time-consuming and error-prone for many screen readers users. We introduce *GestureCalc*, a digital calculator that uses target-free gestures for arithmetic tasks. It allows eyes-free target-less input of digits and operations through taps and directional swipes with one to three fingers, guided by minimal audio feedback. We conducted a mixed methods longitudinal study with eight screen reader users and found that they entered characters with *GestureCalc* 40.5% faster on average than with a typical touch screen calculator. Participants made more mistakes but also corrected more errors with *GestureCalc*, resulting in 52.2% fewer erroneous calculations than the baseline. Over the three sessions in the study, participants were able to learn the *GestureCalc* gestures and efficiently perform short calculations. From our interviews after the second session, participants recognized the effort in learning a new gesture set, yet reported confidence in their ability to become fluent in practice.

CCS Concepts
•**Human-centered computing** → **Accessibility technologies;** •**Hardware** → *Touch screens;* •**Social and professional topics** → People with disabilities;

Author Keywords
Eyes-free entry; gesture input; digital calculator; touch screen; mobile devices.

INTRODUCTION
The digital calculator is a common application that many people use on touch screen devices. This application is generally easy for sighted people to use; they visually locate targets in the form of soft buttons and tap them to get a desired result. Buttons correspond to digits (i.e., 0-9), the decimal point and operations (e.g., subtraction, multiplication, backspace, equals). For people who use screen readers (e.g., VoiceOver for iOS, TalkBack for Android), finding and activating buttons in a spatial layout can be time consuming. The purpose of this paper is to explore a design for a calculator that uses eyes-free target-less gestures, eliminating the need to explore for a button when using a screen reader. We call this *GestureCalc*, an eyes-free gesture-based calculator for touch screens. Eyes-free interaction has been explored in a variety of research projects (e.g., [5, 30, 34, 18, 4, 12]). The authors in [24] have also uncovered differences in the types of gestures sighted versus blind people find intuitive. Eyes-free solutions for accessible numeric input include Tapulator (gesture-based numeric input) [36], DigiTaps (minimal audio feedback for numeric input) [3], and BrailleTap (Braille-based gesture calculator) [1]. Our goal is to combine the advantages and address the disadvantages of existing solutions to create a single gesture-based calculator application usable for screen reader users, typically visually impaired users.

(a) *GestureCalc* (b) Baseline

Figure 1. (a) Performing a three-finger tap on *GestureCalc* our novel eyes-free app with target-free rich gestures, versus (b) typing "5" on *ClassicCalc* (a typical touch screen calculator), the baseline.

To create *GestureCalc* we modified DigiTaps[1.8] [3] for digits and defined new metaphor-based gestures for basic calculator operations. The overall goal is to improve the accessibility of state-of-the-art digital calculator applications by: 1) designing gesture codes based on conceptual metaphors and 2) requiring one or two simple gestures for each digit or operation. We

*The first two authors contributed equally to this work.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ASSETS '19, October 28–30, 2019, Pittsburgh, PA, USA.
© 2019 Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-6676-2/19/10 ...$15.00.
http://dx.doi.org/10.1145/3308561.3353783

evaluated our prototype in a longitudinal study with eight participants over three sessions. During each session we evaluated: 1) the rate of error corrections in the input, 2) the speed of entering the gestures, and 3) the memorability and intuitiveness of the gestures.

Our primary contributions are:

- We designed a novel eyes-free target-less digital calculator application that uses a minimal number of accessible gestures to enter digits and operations. Our code is available online[1] and we also plan to release the app.
- We proposed intuitive gestures based on conceptual metaphors that are both memorable and easily learnable for visually impaired people as suggested by our study.
- We conducted a mixed methods study where participants performed mathematical calculations, in order to evaluate the effectiveness of *GestureCalc* for screen reader users. We found that participants entered calculations 40.5% faster and performed 52.2% fewer erroneous calculations than with the baseline.

RELATED WORK

Gestures and Metaphors

Modern understanding of metaphors goes beyond substitution and comparison in language [42, 16]. In cognitive linguistics, Lakoff and Johnson's theory of conceptual metaphors [25] states that we understand new knowledge by "mapping from known domains to unknown domains" through recurring structures within our cognitive processes (e.g., image schemas) or deeper conceptualizations based in our physical, bodily experience. Though these conceptualizations are influenced by our socio-cultural environment and unique physical experiences, many schemas are grounded in experiences that transcend culture, such as forward motion, occlusion, containment, sensation, and perception. [8, 28].

Metaphors are commonly used in the HCI community. Hurtienne and Blessing [22] found schema-aligned interface elements to be more intuitive to users than interface elements that reversed the established schema. Loeffler et al. isolated schemas in user interview transcripts and mapped these to their designed interface [27]. Hiniker et al. found users were more efficient in working with metaphorical visualizations that aligned with documented image schemas [20]. Finally, Kane et al. found symbol-based gesture languages to be less intuitive than metaphor-based languages for blind people [24]. Given the potential of metaphors to improve usability for blind people, we designed *GestureCalc* to use metaphoric input instead of symbolic input. Gestures with directional swipes are grouped according to whether they increase (e.g., multiplication, addition) or decrease (e.g., division, subtraction) a value and are oriented according to the metaphor "More is up" (e.g., addition is an upward swipe).

Eyes-free Entry

Several researchers have developed eyes-free text entry systems for touch screen devices (e.g., [5, 30, 34, 4, 12, 41]). Touch screen operating systems generally have default screen readers, such as Apple's VoiceOver for iOS [2] or Android's TalkBack [15], which use interaction techniques introduced in SlideRule [23]. Screen reader based text entry guides people through audio feedback to search for targets with one or more fingers on the screen and then perform a second gesture such as a split-tap to select that character to input, a process that can be cumbersome. Bonner et al. developed No-Look-Notes [9], a soft keyboard dividing character input into a first split-tap to select from 8 segments of a pie menu and a second split-tap to select a single character from that segment. This makes character selection easier and faster than with a QWERTY soft keyboard with screen reader, where buttons are very small. However, this still requires searching and targeting, which results in slow character input rates. Speech input is a possible alternative for eyes-free entry, but it has several limitations: 1) it cannot be used in quiet environments, 2) the input is prone to error, especially in noisy environments, and 3) it is not a feasible solution for speech-impaired users.

Eyes-free entry techniques using Braille have also been used to improve touch screen accessibility for blind people (e.g., [34, 30, 6]). Alnfiai et al. proposed BrailleTap [1], a calculator application that uses taps in the form of Braille patterns for numeric input together with other gestures for calculator operations. However, Braille-based inputs have several drawbacks. First, these techniques require multiple gestures per character input (i.e., up to six), which leads to a low input speed. Second, Braille-based inputs require knowledge of Braille dot patterns. Although Braille was developed for people with visual impairments, a report from the National Federation of the Blind states that only 10% of legally blind people can read Braille [33]. This makes Braille legible to only a small percentage of blind people, but we wanted our app to be accessible to a wider population.

Eyes-free input methods have also been explored for entering digits (e.g., Tap2Count [18], Digitaps [3]) and operations (e.g., Tapulator [36]). Tap2Count allows users to touch an interactive screen with one to ten fingers to enter digits. This requires considerable physical effort and cannot be easily scaled to small touch screen devices like mobile phones. DigiTaps uses an eyes-free gesture language based on a prefix-free coding scheme for numeric input with haptic or audio feedback on touch screen devices. However, both Tap2Count and DigiTaps do not include operations, which increase the challenges of designing usable gestures and implementing a working system. Tapulator extends Digitaps by adding gestures for operations, but the gestures are symbolic, based on the printed structures of the operators, rather than metaphorical, which should hinder learnability and memorability for visually impaired users [24].

Existing Products

The idea of using gestures for calculations dates back to early implementation of a touch screen calculator on a digital watch (Casio AT-550) [11]. More recently, MyScript created a calculator [32] that takes handwritten characters on a touch screen as input for mathematical calculations, but it is not usable by blind users. A handful of gesture-based calculators are also available online (e.g., Sumzy for iPhone [7], Swipe

[1] https://github.com/bindita/GestureCalculator

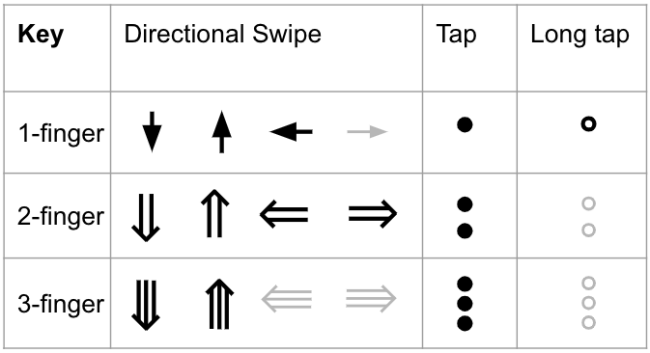

Figure 2. Symbolic representations of a superset of our gestures, with a visual shortcut for each gesture. Gestures currently used in *GestureCalc* are marked in black, while gestures currently unused are marked in grey.

Calculator for Android [31] and Rechner Calculator [35]). All of these calculators have a 3×3 keypad for the digits, and only the operations are performed using tap and swipe gestures on or above the keypad area on the screen. To the best of our knowledge, our calculator is the first application which is free of any buttons or targeted gestures.

DESIGN

We build upon the digit set introduced in DigiTaps [3], adding gestures for basic arithmetic operations. Our application also improves character entry speed compared to a classic calculator application by: 1) allowing users to interact with any part of the screen (i.e., target-less interaction), 2) requiring a maximum of two gestures for every input, 3) avoiding symbolic gestures based on printed characters [24], 4) avoiding complex gestures by using only swipes and taps with up to three fingers, and 5) favoring intuitive gestures based on conceptual metaphors such as "up" for increasing and "down" for decreasing [25].

Gestures

Common gestures for interacting with touch screen devices include tap, swipe, pinch, shake, and rotate. In our design, we only use taps and swipes because they have been found to be easiest to perform and accessible to blind users [24]. Our swipe gestures are directional: up, down, left, and right. We also use a variation of the tap gesture called long tap, in which users press and hold a finger against the touch screen for a short duration. We provide haptic feedback to the user from our app to indicate when the tap is held long enough (0.5 seconds) to be recognized as a long tap gesture. All gestures can be performed anywhere on the screen with one, two, or three fingers simultaneously. We restrict each gesture to involve a maximum of three fingers because interaction with the 'pinky' finger is difficult [36]. Figure 2 shows the symbolic representations of the different possible gesture inputs in our design.

Character Codes

We define the term 'characters' as the digits 0 to 9 and operations that *GestureCalc* accepts as input. Each character is encoded by a combination of one or two gestures for fast entry. Our character codes are prefix-free (i.e., no character's code is a prefix of another code), which allows unambiguous parsing of input. In addition, our gestures are based on conceptual metaphors that help in remembering the character codes.

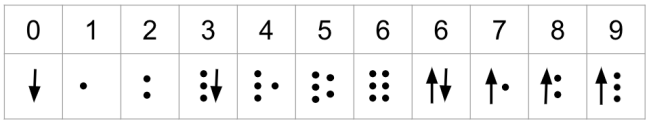

Figure 3. Codes for entering digits.

Digits

We designed 10 different codes to represent the digits 0 to 9, similar to DigiTaps[1.8]. Digit 0 is represented by a one-finger downward swipe, digit 1 by a one-finger tap, and digit 2 by a two-finger tap. The digits 3, 4, 5, and 6 (i.e., the 3 block) can be represented as (3 + 0), (3 + 1), (3 + 2) and (3 + 3) respectively, hence they are encoded by a three-finger tap followed by a one-finger downward swipe, a one-finger tap, two-finger tap and three-finger tap respectively. The delimiter of 0 at the end of digit 3 ensures prefix-free property.

In Digitaps[1.8] [3], digits 7, 8 and 9 are expressed as (10 - 3), (10 - 2), and (10 - 1) respectively. This is inconsistent with our additive scheme for code design. We therefore updated the codes for 7, 8, and 9 to be more semantically similar to 4, 5, and 6, using addition rather than subtraction. We denote digits 6, 7, 8, and 9 (i.e., the 6 block) as (6 + 0), (6 + 1), (6 + 2), and (6 + 3) respectively and represent the prefix 6 for these digits using a one-finger upward swipe. Hence, 6, 7, 8, and 9 are represented by a one-finger upward swipe followed by a one-finger downward swipe, one-finger tap, two-finger tap, or three-finger tap respectively. Note that the digit 6 has two different representations. Figure 3 shows the visual representation of the codes for entering digits using visual shortcuts introduced in Figure 2.

Our coding scheme uses an average of 1.7 gestures per digit, which is fewer than 1.8 gestures for Digitaps[1.8] and 2.5 gestures for BrailleTaps. The trade-off as compared to Digitaps[1.8] is that we use directional swipes. *GestureCalc* digit codes are therefore slightly more complex, but our pilot study offered evidence that they are still easily learnable. This suggests that the increased benefit for fast character entry may offset the cost of additional complexity.

Operations

We oriented directional swipes used in *GestureCalc* operations according to a conceptual metaphor that "more is higher" [25]. For instance, addition increases the value of a number, so we represent '+' operation with a two-finger upward swipe. Subtraction, on the contrary, decreases a number's value and hence '-' operation is represented by a two-finger downward swipe. Our gesture for '-' operation can be used as either an operator between two operands or to negate a single operand. Multiplication implicitly means multiple additions, hence '*' operation is represented by a three-finger upward swipe. Similarly, division is multiple subtractions (and the inverse of multiplication), hence '/' operation is represented by a three-finger downward swipe. Finally, the '.' (decimal point) operation is represented by a long tap.

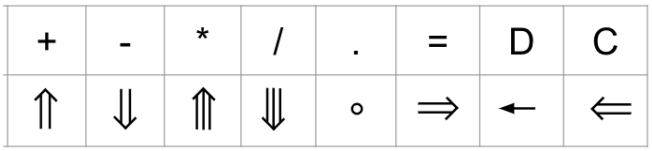

Figure 4. Codes for entering operations.

The '=' (equals) operation metaphorically moves the expression forward by generating a result. Hence, it is represented by a two-finger horizontal swipe from left to right (i.e., two-finger right swipe). Incidentally, this also resembles the shape of the equals symbol. When the user enters the equals operation, the application displays and speaks the computation's result and clears the input for the next computation. The 'D' (delete) operation deletes one character at a time and speaks the character being deleted. In left-to-right writing systems such as Braille [44] or written English, backspace conventionally deletes a character to the left of the cursor. We therefore represent this with a one-finger left swipe. The 'C' (clear) operation deletes all characters in the input, which is equivalent to multiple deletions, so it is represented by a two-finger left swipe. Figure 4 shows the visual representation of the codes for entering operations.

Formative Pilot Study

We conducted a pilot study to evaluate the memorability and intuitiveness of our gesture codes and to improve the usability and functionality of our prototype application. We recruited four participants (one of whom was blind) and conducted two 30-minute sessions (separated by 2 or 3 days), each having two tasks. The first session started with a brief training and practice period, the second session directly started with the practice period. During task 1, participants were asked to verbally describe the gesture code for each character in a random order. The error rate (percentage of wrong answers) in recalling codes decreased to 0 in session 2 for all participants, suggesting that our gesture language is easily memorable. During task 2, the participants were asked to enter a series of 15 random expressions of varying length. All participants achieved a 3% or less error rate in every session, suggesting that the gesture set and the app have a good level of usability. The rate of character entry increased by 24% from session 1 to session 2, indicating speed improves with practice.

We observed it was difficult for participants to remember and enter the entire prompted expression. They often asked for repetitions, which affected character entry speed. To avoid this, we reduced the overall length of expressions for the main study and read the expressions in parts. During post-pilot interviews, participants mentioned that *GestureCalc*'s grouping of digits and operators to share common types of gestures helped. Participants had difficulty remembering the digit 6 was two consecutive three-finger taps, whereas 6 in denoting digits 7, 8, 9 is a one-finger upward swipe. We mitigated this confusion by overloading the code for 6 to include both $3+3$ and $6+0$ (i.e., as described in Character Codes). Finally, our blind participant suggested reading deleted characters aloud.

Additional Features

GestureCalc provides audio feedback (i.e., speaks the entered character) after every digit or operator, similar to Digitaps. Digitaps also used different types of haptic feedback for different gestures as an alternative to audio feedback and found that haptic feedback results in faster input. However, haptic feedback was found to have a higher error rate than audio feedback. Error recovery in long calculations is more costly than in simple number entry, because it requires users to restart the expression from the beginning. Additionally, providing multiple types of haptic feedback for our diverse gesture set may confuse users. Our implementation currently supports mathematical calculations involving just two operands. A readback feature enables users to shake the device to trigger audio feedback, reading aloud the expression that has been entered. This feature is designed to be helpful for feedback while entering expressions involving long operands.

EVALUATION METHOD

We conducted an IRB-approved study to evaluate the design and implementation of our application. We describe the study methods and outcomes in the following subsections.

Participants

We recruited 8 participants for this study, 6 male and 2 female, ranging in age from 23 to 58. Our inclusion criterion was answering "yes" to the question "When you use a touch screen, do you typically use a screen reader?" One participant used an Android device as their primary touch screen device, all others used Apple devices. All participants except one reported using a touch screen every day. Most participants said they use a calculator at least once a week at home / office / classroom / public places for different activities such as calculating tips, unit conversions, and personal budgeting. Their most common calculations (e.g. addition, multiplication, taking average, calculating percentages) do not require a scientific calculator. Table 1 describes individual participant mobile device and calculator use in detail. We compensated participants with US $80 over three sessions and reimbursement for travel expenses.

Apparatus and Conditions

GestureCalc is developed for iOS using the Xcode platform on Mac and the Swift programming language. For the study, we installed it on an iPhone 7 with touch screen dimensions of 5.44×2.64 inches. We allowed both portrait and landscape modes for using the application.

Most applications in prior research require some target-dependent input, hence we compared *GestureCalc* to the default iOS calculator, which we call *ClassicCalc*. To accurately measure performance and add detailed logging, we recreated the default iOS calculator with the same button locations, sizes, and audio labels. However, we removed the '%' button and added the functionality of the '+/-' button to the '-' button, to be consistent with our *GestureCalc* implementation. In *ClassicCalc*, participants type digits by using the standard eyes-free typing mode of iOS, which involves seeking with VoiceOver audio guidance, then performing a double-tap or split-tap to activate the selected

PID	Age	Gender	Primary touch screen device	Primary mobile input method	Primary mobile output method	Mobile device use frequency	Primary calculator	Calculator frequency
P1	27	M	iPhone	braille keyboard	braille display	daily	Voice assistants	a few times a week
P2	39	M	Key One Blackberry	phone's tactile keyboard	TalkBack	daily	Windows 10 calculator	at least a couple times a month
P3	42	F	iPhone	braille screen input	VoiceOver	daily	Windows calculator	every day at work
P4	46	M	iPad	virtual keyboard w/ magnification, color inversion	magnification, color inversion, VoiceOver	daily	iOS calculator with hand lens	once or twice a week
P5	23	F	iPhone	braille screen input	VoiceOver	daily	iOS calculator	several times a week
P6	58	M	iPhone	touch typing	VoiceOver	daily	Siri, Windows calculator, iOS calculator	daily
P7	27	M	iPhone	braille screen input	VoiceOver	daily	Python console	weekly or every few weeks
P8	46	M	iPhone	touch typing	VoiceOver	weekly	Windows calculator	several times a week

Table 1. Participant demographics and summary of mobile device and calculator use. PID denotes participant ID.

key. The only target-free calculator we are aware of is BrailleTap [1], but we did not compare with it because we did not require Braille literacy for study participants.

Procedure

We conducted three 1-hour sessions with each participant, with each pair of consecutive sessions separated by at least 4 hours but not more than 57 hours for a given participant, as in [29]. Each participant used both *GestureCalc* and *ClassicCalc* in every session, counterbalanced so that half of participants used *GestureCalc* first in their first and third sessions, while the rest used *GestureCalc* first in their second session.

In the first session, participants went through a learning period followed by a testing period for each app. The *GestureCalc* learning period started with a facilitator describing each gesture and giving the participants a chance to perform the gesture once. The participants were then asked to type "practice sequences" consisting of the gestures that had just been learned. Practice sequences included "012345689.", "CD", and "+-*/=". Each participant was asked to type each practice sequence three times. The *ClassicCalc* learning period started with a facilitator describing the spatial layout of the classic calculator. The participant was asked to type the same practice sequences as with *GestureCalc*. During the learning period for their first app, each participant was asked to set their preferred VoiceOver speed so that their performance during testing would not be affected due to an uncomfortably fast or slow VoiceOver speed. The same speed was then used for all their testing periods throughout the three sessions.

The testing period for each app consisted of a series of trials. In each trial, the participant was given an arithmetic expression or computation to enter into the calculator. An example of a trial is given below:

Desired input: 72 + 58 =
Transcribed input: 73D2 /D+ 59 =
Final input: 72 + 59 =

Each expression was 4 to 6 characters long and had one of the following two forms:

a) <operand> <operator> <operand> <equals>
b) <operand> <clear>

Each entity (enclosed within <>) was prompted separately from a laptop, allowing the participant to enter the entity on the mobile device before the next entity is prompted from the laptop. This helped the participants to easily remember the prompts while entering them.

The testing period started with 5 unrecorded warm-up trials. Warm-up trials were followed by three blocks of 10 recorded trials. For the recorded trials, participants were requested to "type as quickly and accurately as you can". Expressions in the trials were generated randomly, but we ensured the same frequencies across digits and across operators within each block. Each participant was given the same set of expressions in the same order during session 1. During the recorded trials, we recorded a time stamp at the beginning and end of each prompt, gesture (for *GestureCalc*), button press (for *ClassicCalc*), and audio feedback. To calculate the total time for a trial, we subtracted the time taken to prompt the expressions so as to only count time taken for character entry.

The second session consisted of a testing period for each app followed by an interview. The testing period was conducted exactly as in session 1, except with a different set of expressions for the recorded trials. During the warm-up trials, participants had a chance to re-familiarize themselves with the app and ask for clarifications if needed. Interview methodology is described in the following subsection.

In the third session, there was a testing period followed by a NASA Task Load Index (TLX) [17] for each calculator. The recorded trials used a different set of expressions from session 1 or session 2. For the TLX, the facilitator asked the participants six questions to rate the workload of the application after using each calculator, based on their use of that application in session 3 only.

Interview Methodology

Interviews were semi-structured, organized around five research questions: 1) What are participants' current text input techniques? 2) What are participants' current calculator needs? 3) What are the trade-offs between *GestureCalc*and other calculators? 4) What improvements can we make to *GestureCalc*? and 5) What would it take for *GestureCalc* to be adopted? Interviews were conducted by one author in English and lasted about 30 minutes.

Interview transcripts and interviewer notes were then coded and organized via thematic analysis [10]. Interviews were analyzed by the interviewer and a second author. First, they both independently coded one interview and then discussed discrepancies to come to a shared understanding of the initial codes. Each remaining interview was coded by a single author. Our six initial codes (not to be confused with *GestureCalc*'s character codes) were: a code for each of the five research questions, a code for "metaphor" motivated by our literature review. Five inductive codes were added during coding (e.g., "guesswork", a term first mentioned by a participant). Codes were applied to arbitrary selections of text.

Study Design and Analysis

Our experiment utilized a $2 \times 3 \times 3 \times 10$ within-subjects design with the following factors and levels:

- *Technique*: *ClassicCalc*, *GestureCalc*
- *Session*: 1-3
- *Block*: 1-3
- *Trial*: 1-10

Factors of particular interest were *Technique* and *Session*, as we were interested in how the two techniques compared and how their performance evolved over the 3 sessions. Within each session, each of the 8 participants used both techniques in a series of 3 blocks of 10 trials each, with short breaks in between. Thus, our study data consisted of $8 \times 2 \times 3 \times 3 \times 10 = 1440$ trials in all. For each trial, we computed characters per second (CPS), uncorrected error rate (UER), a binary value indicating whether there were any errors in the final calculation, and the corrected error rate (CER). Definitions for UER and CER were taken from established text entry research [39].

For our statistical analyses, we used a linear mixed model ANOVA for CPS [14, 26, 43]. For our analysis of UER and CER, which do not conform to the assumptions of ANOVA, we used the nonparametric aligned rank transform procedure [19, 37, 38, 46]. In all of these analyses, *Technique* and *Session* were modeled as fixed effects. *Block* was modeled as a random effect nested within *Session*, and *Trial* was modeled as a random effect nested within *Block* and *Session*. *Participant* was also modeled as a random effect to account for repeated measures. Any effect of VoiceOver speed was considered to be part of the random effect of *Participant*.

RESULTS

This section presents the results of our within-subjects experiment, examining the effects of *Technique* (*GestureCalc* / *ClassicCalc*) and *Session* on characters per second (CPS), uncorrected error rate (UER), number of erroneous calculations (NEC) and corrected error rate (CER).

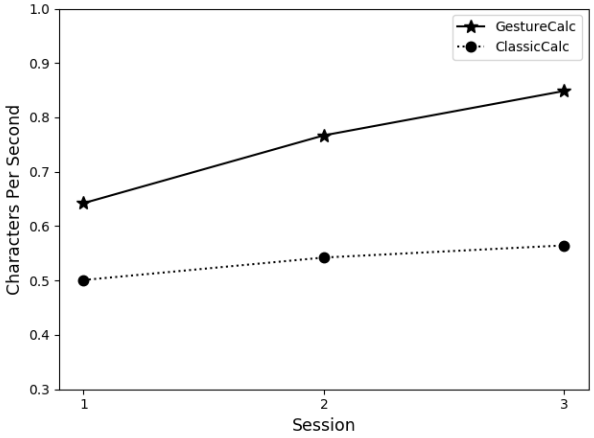

Figure 5. Characters Per Second by *Session* × *Technique*.

Characters Per Second

We examined speed (i.e., rate of character entry) using characters per second (CPS). The average CPS for the *ClassicCalc* was 0.536 (SD=0.150), whereas the average CPS for *GestureCalc* was 0.753 (SD=0.240), a 40.5% speed-up. An omnibus test showed there were significant main effects of *Technique* ($F_{1,1340} = 751.34$, $p < .0001$) and *Session* ($F_{2,6} = 18.24$, $p < .005$) on characters per second. There was also a significant *Technique* × *Session* interaction ($F_{2,1340} = 27.45$, $p < .0001$). Figure 5 shows the characters per second for *GestureCalc* and *ClassicCalc* over each session, averaged over all participants. The graph shows that *GestureCalc* has higher input speed compared to *ClassicCalc* in all three sessions.

Post hoc pairwise comparisons corrected with Holm's sequential Bonferroni procedure [21] reveal that all pairwise comparisons in Figure 5 are significantly different except for *ClassicCalc* session 1 versus 2 ($t_{274} = -1.69$, *n.s.*) and 2 versus 3 ($t_{274} = -0.91$, *n.s.*). *ClassicCalc* improved from session 1 to 3 ($t_{274}) = -2.59$, $p < .05$). *GestureCalc* improved significantly between all sessions: 1 versus 2 ($t_{274} = -5.09$, $p < .0001$), 2 versus 3 ($t_{274} = -3.32$, $p < .01$), and 1 versus 3 ($t_{274} = -8.41$, $p < .0001$). This indicates participants entered characters at a faster rate with more practice with *GestureCalc*, whereas performance for *ClassicCalc* had largely saturated.

Uncorrected Error Rate

Uncorrected error rate (UER) refers to the rate of incorrect characters remaining in the final input. Low UER indicates that participants could reliably and accurately use the calculator application (i.e., receive the correct output). The average UER for *ClassicCalc* is 2.29% (SD=7.84%), whereas the average UER for *GestureCalc* is 0.92% (SD=4.40%), a 59.8% reduction. An omnibus test showed there was a significant main effect of *Technique* ($F_{1,1340} = 11.06$, $p < .001$) on UER. However, we did not find a main effect of *Session* ($F_{2,6} = 2.67$, *n.s.*), nor did we find a significant *Technique* × *Session* interaction ($F_{2,1340} = 1.55$, *n.s.*). This means UER remained similar for both methods over all three sessions.

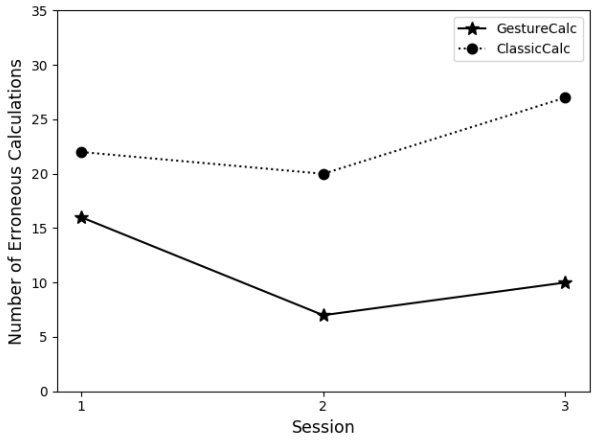

Figure 6. Number of Erroneous Calculations by *Session* × *Technique*.

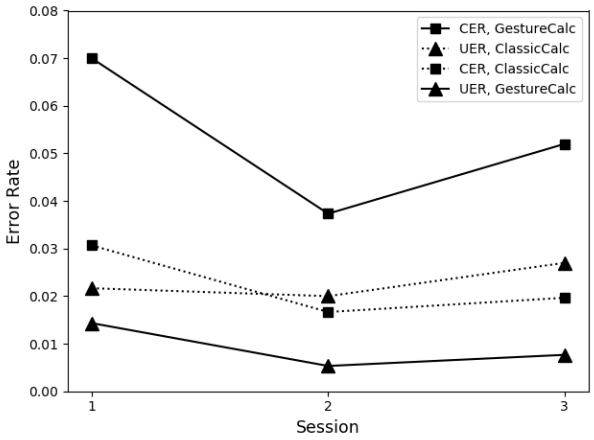

Figure 7. Uncorrected and Corrected Error Rates by *Session* × *Technique*.

PID	GC_{CPS} - CC_{CPS}	CC_{TER} - GC_{TER}	CC_{NEC} - GC_{NEC}
P1	54.1%	-0.013	2
P2	61.4%	-0.028	0
P3	30.3%	-0.031	-3
P4	12.8%	0.034	9
P5	34.8%	-0.067	-5
P6	32.5%	-0.049	1
P7	67.3%	0.045	22
P8	39.3%	-0.029	10

Table 2. Metrics by participant (larger values imply better performance of *GestureCalc* (GC) over *ClassicCalc* (CC). Here CPS is characters per second expressed as a percentage over CPS for CC, TER (Total Error Rate) = UER + CER, and NEC is number of erroneous calculations.

Number of Erroneous Calculations

An 'erroneous calculation' can be defined as any trial that has uncorrected errors in the final input, because such errors would result in incorrect calculations for the user. *ClassicCalc* had 69 (9.58%) erroneous trials whereas *GestureCalc* had only 33 (4.58%) erroneous trials, 52.2% fewer than *ClassicCalc*. Fisher's exact test [13] shows a significant difference in these error proportions in favor of *GestureCalc* ($p < .001$). A second analysis using logistic regression in a generalized linear mixed model [40] shows that *Technique* had a significant effect on likelihood of an erroneous trial ($\chi^2_{(1,N=1440)} = 14.25$, $p < .001$). There was no *Session* main effect ($\chi^2_{(1,N=1440)} = 3.09$, *n.s.*) or *Technique* × *Session* interaction ($\chi^2_{(2,N=1440)} = 2.81$, *n.s.*), corroborating the aforementioned analyses of uncorrected error rate. Figure 6 shows that participants entered more erroneous calculations with *ClassicCalc* than with *GestureCalc* in all three sessions.

Corrected Error Rate

Corrected error rate (CER) refers to the rate of incorrect characters in the transcribed input that were later corrected in the final input. Corrected errors therefore do not adversely affect calculator calculations, but they do take time and attention to fix (i.e., using the delete or clear operators). The average CER for *ClassicCalc* is 2.23% (SD=8.00%), whereas the average CER for *GestureCalc* is 5.31% (SD=9.65%), a 138.1% increase. An omnibus test showed there was a significant main effect of *Technique* ($F_{1,1340} = 27.17$, $p < .0001$) on CER. An omnibus test also showed a main effect of *Session* on CER ($F_{2,6} = 12.67$, $p < .01$). Finally, we found a significant *Technique* × *Session* interaction ($F_{2,1340} = 11.44$, $p < .0001$). The CER values decreased after the first session, but increased after the second session.

Post hoc pairwise comparisons conducted with Wilcoxon signed-rank tests [45] revealed that *GestureCalc*'s CER was lower in session 2 than in session 1 ($p = .080$). By contrast, *ClassicCalc* did not show significant changes in CER over sessions. Within session 1, the two techniques were only marginally different ($p = .107$). By sessions 2 and 3, the two techniques were significantly different ($p < .05$).

Figure 7 shows the UER and CER values using *GestureCalc* and *ClassicCalc* averaged over all participants. We see that participants made more errors while entering expressions using *GestureCalc* compared to using *ClassicCalc*, but were also able to correct the errors more frequently, so that the final inputs using *GestureCalc* were more accurate than using *ClassicCalc*.

Table 2 shows the performance of individual participants based on our metrics, averaged over all trials in the 3 sessions. We note that every participant except P4 had more than 30% faster rate of character entry (CPS) with *GestureCalc* compared to *ClassicCalc*, with negligible difference in the total error rate (sum of UER and CER). All participants except P4 and P7 also entered fewer erroneous calculations with *GestureCalc* compared to *ClassicCalc*. Overall, we found *GestureCalc* was more efficient to use and has a better overall performance compared to *ClassicCalc*.

NASA Task Load Index

NASA Task Load Index (TLX) asks participants to rate the workload of a task on six different scales: mental, physical, temporal, performance, effort, and frustration [17]. We used the Wilcoxon signed-rank test to determine the statistical significance of differences.

The mental demand scale prompted participants with the question, "How mentally demanding was the task?", and

ranged from low (1) to high (20). The average mental demand for *ClassicCalc* was 5.88 (SD=4.64) and for *GestureCalc* was 7.50 (SD=4.47). This difference was not statistically significant (*Z*=-0.63, *n.s.*). The higher mental demand of the *GestureCalc* was due to the learning curve, because participants were familiar with *ClassicCalc* but had to familiarize themselves with *GestureCalc*.

The physical demand scale prompted participants with the question, "How physically demanding was the task?", and ranged from low (1) to high (20). The average physical demand for *ClassicCalc* was 3.13 (SD=2.03) and for *GestureCalc* was 3.25 (SD=3.81). This difference was not statistically significant (*Z*=0.64, *n.s.*).

The temporal demand scale prompted participants with the question, "How hurried or rushed was the pace of the task?", and ranged from low (1) to high (20). The average temporal demand for *ClassicCalc* was 5.13 (SD=5.03) and for *GestureCalc* was 3.63 (SD=3.11). This difference was not statistically significant (*Z*=0.00, *n.s.*).

The performance scale prompted participants with the question, "How successful were you in accomplishing what you were asked to do?", and ranged from perfect (1) to failure (20). The average performance rating for *ClassicCalc* was 4.63 (SD=4.41) and for *GestureCalc* was 5.50 (SD=3.34). This difference was not statistically significant (*Z*=-0.70, *n.s.*). It is important to note that this rating is in line with our observation of higher CER but lower UER of *GestureCalc* compared to *ClassicCalc*. Participants mentioned that they rated their performance more towards failure because they were aware of the mistakes they made during character entry.

The effort scale prompted participants with the question, "How hard did you have to work to accomplish your level of performance?", and ranged from low (1) to high (20). The average effort rating for *ClassicCalc* was 7.00 (SD=4.72) and for *GestureCalc* was 6.38 (SD=4.41). This difference was not statistically significant (Z=0.07, *n.s.*).

INTERVIEW RESULTS

In this section we summarize results from interviews with participants after they completed the second session.

Calculator Use

Our participants regularly perform a variety of calculations, with calculations involving money being more frequent. For personal finances, participants compute tips, monthly expenses, and expenses in the grocery store. At work, participants compute bills and financial estimates. One participant primarily uses a calculator for unit conversion. P5 highlighted the importance of accessible calculators as she said, "*Right now, I'm studying to take the test to get into math classes. I haven't been able to take the test yet because I don't have a decent calculator.*" Indeed, participants often preferred to use a laptop or desktop computer for doing calculations because digits can be typed directly with keyboard buttons, but perhaps such a computer is not allowed in P5's math test.

Device Issues

Physicality of devices and fingers played a role in participant ability to perform gestures accurately. Our phone was too narrow for some participants to make a three-finger tap in portrait mode (the default), so they switched to landscape. P2 (among others) noted it was important to know where the edge of the screen was in order to perform the gestures accurately, saying, "*You had to keep your fingers in the middle of the screen, no matter if you were doing it in landscape or portrait.*" P8 described his own phone case, which has "*a raised rim around both sides, so your fingers don't actually get off the touch area of the screen.*" P2 also pointed out that gesture performance could be affected by sweaty fingers, residue on the screen, or a moving environment such as a bus.

Causes of Errors and Confusion

Participant feedback on the relative difficulty of gestures varied widely across participants, often conflicting. One common thread was that participants found operators relatively intuitive (P1, P6, P7). In fact, one participant accidentally tried swiping to delete while using the classic calculator. The different blocks in the gesture set design for digits caused confusion. Despite disagreement on the relative difficulty of the 3 block and the 6 block, P6 identified context-switching between the blocks as an important hurdle, saying, "*making that transition like from a 5 to a 7, that's a little challenging.*"

Memorability and Mental Demand

The mental demand of remembering the codes was the primary aspect that participants described in their experience using *GestureCalc*, and it was a source of error and confusion. Each participant's first session with *GestureCalc* started with the facilitator teaching them to use it, so they all had similar, structured introductions to the codes. As P8 put it, using *GestureCalc* "*takes a little rearranging of your way of thinking.*" However, participants felt they could learn the codes through practice and repetition.

Like *GestureCalc*, learning Braille also requires memorization. P4 explained, "*It's not a hard learning curve . . . I started to get a little anxious at first, because I thought, 'Oh, no, here we go with Braille again.' But once I just let that thought go . . . No, I wouldn't hesitate to tell anybody to try this.*" For him, learning our codes was not as challenging as learning Braille. P7, on the other hand, suspected that people may not bother learning the codes, saying, "*I wouldn't call this a really steep learning curve, [but] there's a learning curve. If I'm downloading a calculator, I want to be able to just start using it.*" Helping people learn the codes will be an important hurdle in achieving broad impact and adoption of eyes-free gestures.

Feature Suggestions

Indeed, most of the participants identified that new users would need a tutorial or manual to learn the gesture set. It could either be an interactive tutorial, "*similar to [the learning period] that we did in the first session*" (P7), a text-based "*quick-reference*" (P8), or both. We heard a wide variety of suggestions for improving *GestureCalc*, including: 1) additional mathematical operations to support, 2) using the iOS VoiceOver instead of our custom self-voicing (to avoid having to turn off VoiceOver

and to share VoiceOver's settings), and 3) ways to navigate, edit, and read back input or to repeat the output of a calculation. Although we included a readback feature activated by shaking the device, it was rarely used.

Some participants suggested redesigning the gesture set. For example, digit codes could be closely based on the spatial location of each digit key on a phone keypad, incorporating upward swipes for keys on the top row and downward swipes for keys on the bottom row (P3), or numeric gestures could have the user "*trace a number onto the screen*" (P8). P1 participant suggested keeping the operator gestures, but replacing digit gestures with Braille input or a virtual keypad. He also suggested tracing a square on the screen for squaring a number. As a mnemonic device, mapping the squaring operation to the shape of a square seems promising (at least for English-speaking users), though previous work argues that such a shape-based symbolic gesture may be challenging to recognize accurately while supporting blind users [24].

Helpfulness of Metaphors

GestureCalc gesture codes are based on particular metaphors (e.g., "A then B" for gestures within digits means "$A+B$", "up swipes mean increasing"), which we referenced in the training to help participants remember the codes. P6 mentioned that these metaphors were useful for him. He explained, "*The fact that [the codes] are in a way that makes sense—up, more, larger—down, smaller—and those connect well with plus, minus, times, divide. They have projected outcomes that you can relate to easily that makes it user friendly.*" P8 drew attention to the fact that the metaphors we chose are not necessarily universal. Explaining his difficulty with the gestures for 3, 5, and 6, he referred to the uniqueness of his cognition: "*It may have a lot to do with the way I coordinate, my own thought processes, and the way my brain works.*" This reminds us that different gestures, different codes, and different calculators should work better for different people.

Comparisons between Calculator Apps

Comparing *GestureCalc* to *ClassicCalc*, participants found it helpful that with *GestureCalc* they can perform the gesture anywhere on the screen, but they recognized that in return this requires memorizing the gestures. Although participants were generally positive about *GestureCalc*, P8 felt that he was slower with *GestureCalc* because it took "*more movement*" to perform the gestures and P2 felt that he made more mistakes with *GestureCalc*. P5 pointed out the main improvement of *GestureCalc* over other phone calculators when she remarked, "*gestures are good; they take the guesswork out.*" The guesswork involved in using a screen reader is to guess where to move your finger on the screen to find what you want. In the words of P1, *GestureCalc* would improve on this because, "*you wouldn't have to hunt around the screen.*"

Some participants identified a similarity between *GestureCalc* and Braille input systems, as both encode digits and operators symbolically rather than spatially. As P5 put it, "*For me, Braille screen input sped up my typing . . . I think that Gesture Calculator would be the exact same thing. Once people got familiar with how to use it, I think it would speed up calculations.*" Participants who use Braille inputs appreciated *GestureCalc* for this similarity to Braille.

DISCUSSION

In this section we discuss the limitations of our study and issues that arose in the study and interviews.

Limitations

Due to difficulty in recruiting blind participants, we conducted our study with only 8 participants and over only 3 sessions. The limited sample of our study provides good evidence, but results are not fully generalizable nor conclusive. In addition, the counterbalancing of the order of app usage for each participant was not ideal because of the odd number of sessions. Nonetheless, we still observe considerable uniformity in relative performance of individual participants on the two calculators in terms of the metrics we used.

Our *ClassicCalc* app supported standard typing but not touch typing[2], to be consistent across all participants regardless of the typing mode they prefer. Although standard typing is thought to be the more conventional mode of text entry, most of our participants expressed a preference for touch typing, and some said it was faster for them than standard typing. P6 acknowledged a speed-accuracy tradeoff, noting that touch typing is more error prone. Those who preferred touch typing, however, were also familiar with standard typing and appeared quite fluent in the technique.

Scalability: Adding Operations

Although *GestureCalc* supports the most common basic mathematical operations, some participants suggested adding scientific operations. If we extended *GestureCalc*, we would follow participant suggestions to increase the memorization requirements as little as possible. One possibility is a modal approach that does not add unique gestures for each new symbol but instead uses one or more new prefix gestures (e.g., unused gestures in Figure 2) to enable reuse of existing gestures to represent new operators and symbols. Another possibility is introducing an escape gesture to enter a mode where the user can swipe through less common operators and symbols. Such repetitive swiping is a common approach to exploring and selecting from lists with a screen reader. These approaches would leave the existing gesture set as-is, following a design principle of keeping common interactions easy while making less common interactions possible.

Error Rates and Speedup

A few participants (P2, P3, P4) observed that the gesture recognition was very sensitive, often capturing stray accidental touches and interpreting them as gestures. Simple gesture codes such as "1" (one-finger tap) can be typed quickly and easily, but could also easily be generated by accidental touches.

[2]VoiceOver supports two modes of typing, standard typing and touch typing. In standard typing, the user seeks for a key by sliding a finger over the screen, VoiceOver speaks the name of each key as it is touched, and the user performs a double tap or split tap anywhere on the screen to activate last key spoken. In touch typing, the user slides a finger across the screen to seek, VoiceOver reads each key under the finger, and lifting the finger activates the last key touched.

In contrast, *ClassicCalc* is robust to stray touches because keys are activated by double tap or split tap, which is much less likely to occur accidentally, but also makes *ClassicCalc* slower. This speed-accuracy trade-off may have contributed to the higher error rate with *GestureCalc*. Two-gesture digits may have also raised the error rate with *GestureCalc*. For example, consider the situation where a user tries to type "7" (i.e., a one-finger upward swipe followed by a one-finger tap). If the swipe is incorrectly recognized as a tap and the user continues typing, they end up typing "11", generating two errors that both subsequently need to be deleted.

For *ClassicCalc*, the amount of time spent typing can be broken down into think time (deciding what number to type), seek time (finding the key), and gesture time (double or split tap). Because of its target-free design, *GestureCalc* eliminated the need to seek for keys, thereby reducing the app's verbosity. Participants appreciated the low verbosity of *GestureCalc*. P8 said "*The amount of speech [in GestureCalc] was right, it wasn't overly wordy... one of the problems I have with a lot of talking devices is they just go on and on and on.*" The low verbosity of *GestureCalc* might have made it easier for participants to recognize errors made while typing, contributing to the low uncorrected error rate of *GestureCalc*. Furthermore, the elimination of seek time (one participant expressed frustration at having to seek for the delete key with *ClassicCalc*) and intuitiveness of the delete gesture may have encouraged error correction with *GestureCalc*, further reducing uncorrected error rates. Though the rich gestures of *GestureCalc* may take longer than the simple taps of *ClassicCalc*, we believe that much of the *GestureCalc* speedup can be attributed to elimination of seek time.

Design and Implementation

Out of all the entries for the digit 6, 57% were in the form 3+3 and 43% were in the form 6+0, indicating that having two possible representations allowed participants to choose whichever they remembered and validating our design decision to include both. Participants found three-finger gestures difficult to perform compared to one- or two-finger gestures. This was often because it was difficult for them to synchronize three differently sized fingers to touch the screen at the same time or to fit three fingers within the limited screen width. 2 out of 8 participants preferred landscape mode compared to portrait mode because of additional screen space. Although landscape mode helped with three-finger taps and horizontal swipes, portrait mode was preferred for vertical swipes.

Impact

Potential use cases for *GestureCalc* are similar to those for mobile phone calculators for sighted people: for people without PCs, when in public and/or noisy areas (P7), calculating tips (P8), shopping (P6), or at their computer when they have too many windows open (P6). Although any mobile phone calculator app has limitations (e.g., less powerful than a desktop computer for calculations), *GestureCalc* supports the most common use cases. Furthermore, our contribution extends beyond the app to include our exploration of target-free metaphorical gestures for future applications.

GestureCalc employs a *rich gesture* set, with target-free gestures conveying a command through the gesture itself. In contrast, *ClassicCalc* and many other touch-based apps use simple targeted gestures, where the command is conveyed through target location. *GestureCalc* demonstrates the utility of rich metaphorical gestures in target-free design, an approach that can help make designs accessible to a greater population of users. Apple's iOS already takes advantage of rich gestures in its VoiceOver screen reader, making it difficult to implement rich gesture apps that are compatible with VoiceOver because some of the most desirable rich gestures already activate VoiceOver features. One potential direction is for gesture-based apps to register their gestures with VoiceOver, thereby disabling those gestures for VoiceOver while a registered app is active.

CONCLUSION AND FUTURE WORK

We designed a basic digital calculator application that allows eyes-free target-free input of digits and operations on touch screen devices. Our input is based on simple gestures like taps and swipes with one, two, or three fingers and the coding scheme is based on conceptual metaphors to improve the intuitiveness and memorability. We conducted a study with blind participants to determine the usability, learnability, and memorability of our gesture set design and implementation. Participants entered characters significantly faster and made fewer erroneous calculations compared to a baseline, *ClassicCalc*. Future research could further compare the performance of our application with a wider variety of calculator designs. Our current application also has room for improvement, such as extending its capabilities (e.g., additional operations) and considering additional gestures (e.g., screen-edge gestures [24]).

ACKNOWLEDGMENTS

We would like to thank our participants for the pilot study and the main study for devoting their time and effort in our study. We would also like to thank the reviewers for providing helpful comments. This work was funded in part by the National Science Foundation under award IIS-1702751 and Grant No. CNS-1539179. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1762114. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Mrim Alnfiai and Srinivas Sampalli. 2017. BrailleTap: Developing a Calculator Based on Braille Using Tap Gestures. In *Universal Access in Human–Computer Interaction. Designing Novel Interactions*. Springer International Publishing, Cham, 213–223. DOI: http://dx.doi.org/10.1007/978-3-319-58703-5_16

[2] Apple, Inc. 2018. iPhone Accessibility. (2018). https://www.apple.com/accessibility/iphone/vision/

[3] Shiri Azenkot, Cynthia L. Bennett, and Richard E. Ladner. 2013. DigiTaps: Eyes-free Number Entry on

Touchscreens with Minimal Audio Feedback. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST '13)*. 85–90. DOI:http://dx.doi.org/10.1145/2501988.2502056

[4] Shiri Azenkot, Richard E. Ladner, and Jacob O. Wobbrock. 2011. Smartphone Haptic Feedback for Nonvisual Wayfinding. In *The Proceedings of the 13th International ACM SIGACCESS Conference on Computers and accessibility (ASSETS '11)*. ACM, 281–282. DOI: http://dx.doi.org/10.1145/2049536.2049607

[5] Shiri Azenkot, Kyle Rector, Richard E. Ladner, and Jacob O. Wobbrock. 2012a. PassChords: Secure Multi-touch Authentication for Blind People. In *Proceedings of the 14th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '12)*. ACM, 159–166. DOI: http://dx.doi.org/10.1145/2384916.2384945

[6] Shiri Azenkot, Jacob O. Wobbrock, Sanjana Prasain, and Richard E. Ladner. 2012b. Input Finger Detection for Nonvisual Touch Screen Text Entry in Perkinput. In *Proceedings of Graphics Interface 2012 (GI '12)*. 121–129.

[7] BIHA Studio. 2013. Sumzy for iPhone. (2013). https://www.youtube.com/watch?v=pHCoEIk_cB4

[8] Frank Boers. 1999. When a Bodily Source Domain Becomes Prominent: The Joy of Counting Metaphors in the Socio-economic Domain. *Amsterdam Studies in the Theory and History of Linguistic Science Series 4* (1999), 47–56.

[9] Matthew N. Bonner, Jeremy T. Brudvik, Gregory D. Abowd, and W. Keith Edwards. 2010. No-look Notes: Accessible Eyes-free Multi-touch Text Entry. In *International Conference on Pervasive Computing (PerCom '10)*. Springer, 409–426. DOI: http://dx.doi.org/10.1007/978-3-642-12654-3_24

[10] Virginia Braun and Victoria Clarke. 2006. Using Thematic Analysis in Psychology. *Qualitative research in psychology* 3, 2 (2006), 77–101. DOI: http://dx.doi.org/10.1191/1478088706qp063oa

[11] Bill Buxton. 1984. Touch Screen Calculator Watch. (1984). https://www.youtube.com/watch?time_continue=1&v=UhVAsqhfhqU

[12] James Clawson, Kent Lyons, Thad Starner, and Edward Clarkson. 2005. The Impacts of Limited Visual Feedback on Mobile Text Entry for the Twiddler and Mini-qwerty Keyboards. In *Proceedings of the 9th IEEE International Symposium on Wearable Computers (IWSC '05)*. IEEE, 170–177. DOI: http://dx.doi.org/10.1109/ISWC.2005.49

[13] Ronald A. Fisher. 1922. On the Interpretation of χ^2 from Contingency Tables, and the Calculation of P. *Journal of the Royal Statistical Society* 85, 1 (1922), 87–94. DOI:http://dx.doi.org/10.2307/2340521

[14] Brigitte N. Frederick. 1999. Fixed-, Random-, and Mixed-Effects ANOVA Models: A User-Friendly Guide for Increasing the Generalizability of ANOVA Results. (1999).

[15] Google, LLC. 2018. Android Accessibility Suite. (2018). https://play.google.com/store/apps/details?id=com.google.android.marvin.talkback

[16] Anne Hamilton. 2000. Metaphor in Theory and Practice: The Influence of Metaphors on Expectations. *ACM Journal of Computer Documentation* 24, 4 (2000), 237–253. DOI: http://dx.doi.org/10.1145/353927.353935

[17] Sandra G. Hart and Lowell E. Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In *Human Mental Workload*, Peter A. Hancock and Najmedin Meshkati (Eds.). Advances in Psychology, Vol. 52. North-Holland, 139 – 183. DOI: http://dx.doi.org/10.1016/S0166-4115(08)62386-9

[18] Tobias Hesselmann, Wilko Heuten, and Susanne Boll. 2011. Tap2Count: numerical input for interactive tabletops. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS '11)*. ACM, 256–257. DOI: http://dx.doi.org/10.1145/2076354.2076403

[19] James J. Higgins and S. Tashtoush. 1994. An Aligned Rank Transform Test for Interaction. *Nonlinear World* 1, 2 (1994), 201–211.

[20] Alexis Hiniker, Sungsoo Hong, Yea-Seul Kim, Nan-Chen Chen, Jevin D West, and Cecilia Aragon. 2017. Toward the Operationalization of Visual Metaphor. *Journal of the Association for Information Science and Technology* 68, 10 (2017), 2338–2349. DOI: http://dx.doi.org/10.1002/asi.23857

[21] Sture Holm. 1979. A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics* (1979), 65–70.

[22] Jörn Hurtienne and Luciënne Blessing. 2007. Design for Intuitive Use - Testing Image Schema Theory for User Interface Design. In *Proceedings of the 16th International Conference on Engineering Design (ICED '07)*, Vol. 7. Citeseer, 1–12.

[23] Shaun K. Kane, Jeffrey P. Bigham, and Jacob O. Wobbrock. 2008. Slide Rule: Making Mobile Touch Screens Accessible to Blind People Using Multi-touch Interaction Techniques. In *Proceedings of the 10th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '08)*. ACM, 73–80. DOI:http://dx.doi.org/10.1145/1414471.1414487

[24] Shaun K. Kane, Jacob O. Wobbrock, and Richard E. Ladner. 2011. Usable Gestures for Blind People: Understanding Preference and Performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. 413–422. DOI: http://dx.doi.org/10.1145/1978942.1979001

[25] George Lakoff and Mark Johnson. 1980. The Metaphorical Structure of the Human Conceptual System. *Cognitive science* 4, 2 (1980), 195–208. DOI: http://dx.doi.org/10.1207/s15516709cog0402_4

[26] R. C. Littell, P. R. Henry, and C. B. Ammerman. 1998. Statistical Analysis of Repeated Measures Data Using SAS Procedures. *Journal of Animal Science* 76, 4 (1998), 1216–1231. DOI: http://dx.doi.org/10.2527/1998.7641216x

[27] Diana Loeffler, Anne Hess, Andreas Maier, Joern Hurtienne, and Hartmut Schmitt. 2013. Developing Intuitive User Interfaces by Integrating Users' Mental Models into Requirements Engineering. In *Proceedings of the 27th International BCS Human Computer Interaction Conference*. British Computer Society, 15.

[28] Diana Löffler, Klara Lindner, and Jörn Hurtienne. 2014. Mixing Languages': Image Schema Inspired Designs for Rural Africa. In *Proceedings of the Extended Abstracts of the 32nd Annual ACM Conference on Human Factors in Computing Systems (CHI '14)*. ACM, 1999–2004. DOI:http://dx.doi.org/10.1145/2559206.2581356

[29] I. Scott MacKenzie and Shawn X. Zhang. 1999. The Design and Evaluation of a High-performance Soft Keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '99)*. 25–31. DOI:http://dx.doi.org/10.1145/302979.302983

[30] Sergio Mascetti, Cristian Bernareggi, and Matteo Belotti. 2011. TypeInBraille: A Braille-based Typing Application for Touchscreen Devices. In *The Proceedings of the 13th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '11)*. ACM, 295–296. DOI: http://dx.doi.org/10.1145/2049536.2049614

[31] Mehmed Mert and Ergün Kayis. 2013. Swipe Calculator for Android. (2013). http://www.swipecalculator.com/

[32] MyScript. 2013. MyScript Calculator 2. (2013). https://www.myscript.com/calculator/

[33] National Federation of the Blind. 2009. The Braille Literacy Crisis in America. (2009).

[34] João Oliveira, Tiago Guerreiro, Hugo Nicolau, Joaquim Jorge, and Daniel Gonçalves. 2011. BrailleType: Unleashing Braille Over Touch Screen Mobile Phones. In *IFIP Conference on Human-Computer Interaction*. Springer, 100–107. DOI: http://dx.doi.org/10.1007/978-3-642-23774-4_10

[35] Rechner. 2012. Rechner Calculator. (2012). http://rechner-app.com/

[36] Vaspol Ruamviboonsuk, Shiri Azenkot, and Richard E. Ladner. 2012. Tapulator: A Non-visual Calculator Using Natural Prefix-free Codes. In *Proceedings of the 14th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '12)*. 221–222. DOI:http://dx.doi.org/10.1145/2384916.2384963

[37] K. C. Salter and R. F. Fawcett. 1985. A Robust and Powerful Rank Test of Treatment Effects in Balanced Incomplete Block Designs. *Communications in Statistics-Simulation and Computation* 14, 4 (1985), 807–828. DOI: http://dx.doi.org/10.1080/03610918508812475

[38] K. C. Salter and R. F. Fawcett. 1993. The ART Test of Interaction: A Robust and Powerful Rank Test of Interaction in Factorial Models. *Communications in Statistics-Simulation and Computation* 22, 1 (1993), 137–153. DOI: http://dx.doi.org/10.1080/03610919308813085

[39] R. William Soukoreff and I. Scott MacKenzie. 2003. Metrics for Text Entry Research: An Evaluation of MSD and KSPC, and a New Unified Error Metric. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03)*. ACM, 113–120. DOI: http://dx.doi.org/10.1145/642611.642632

[40] Robert Stiratelli, Nan Laird, and James H. Ware. 1984. Random-Effects Models for Serial Observations with Binary Response. *Biometrics* (1984), 961–971. DOI: http://dx.doi.org/10.2307/2531147

[41] Hussain Tinwala and I. Scott MacKenzie. 2009. Eyes-free Text Entry on a Touchscreen Phone. In *2009 IEEE Toronto International Conference Science and Technology for Humanity (TIC-STH '09)*. IEEE, 83–88. DOI:http://dx.doi.org/10.1109/TIC-STH.2009.5444381

[42] Amy Vidali. 2010. Seeing What We Know: Disability and Theories of Metaphor. *Journal of Literary & Cultural Disability Studies* 4, 1 (2010), 33–54. DOI: http://dx.doi.org/10.1353/jlc.0.0032

[43] Brady T. West. 2009. Analyzing Longitudinal Data with the Linear Mixed Models Procedure in SPSS. *Evaluation & the Health Professions* 32, 3 (2009), 207–228. DOI: http://dx.doi.org/10.1177/0163278709338554

[44] Wikipedia. 2018. Braille. https://en.wikipedia.org/wiki/Braille. (2018).

[45] Frank Wilcoxon. 1945. Individual Comparisons by Ranking Methods. *Biometrics bulletin* 1, 6 (1945), 80–83. DOI: http://dx.doi.org/10.1007/978-1-4612-4380-9_16

[46] 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)*. ACM, 143–146. DOI: http://dx.doi.org/10.1145/1978942.1978963