BackSwipe: Back-of-device Word-Gesture Interaction on Smartphones

Wenzhe Cui
Stony Brook University
wecui@cs.stonybrook.edu

Suwen Zhu
Grammarly, Inc.
suwzhu@cs.stonybrook.edu

Zhi Li
Stony Brook University
zhili3@cs.stonybrook.edu

Zheer Xu
Dartmouth College
zheer.xu.gr@dartmouth.edu

Xing-Dong Yang
Dartmouth College
xing-dong.yang@dartmouth.edu

IV Ramakrishnan
Stony Brook University
ram@cs.stonybrook.edu

Xiaojun Bi
Stony Brook University
xiaojun@cs.stonybrook.edu

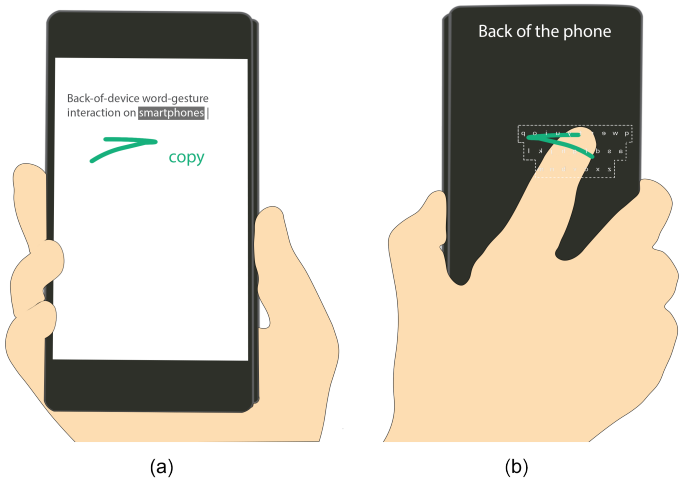


Figure 1: Demonstration of BackSwipe. The user is holding the smartphone with one hand, and uses the index finger to input the word-gesture of the command *"copy"* on the back of the device. The command is triggered on the composing text.

ABSTRACT

Back-of-device interaction is a promising approach to interacting on smartphones. In this paper, we create a back-of-device command and text input technique called BackSwipe, which allows a user to hold a smartphone with one hand, and use the index finger of the same hand to draw a word-gesture anywhere at the back of the smartphone to enter commands and text. To support BackSwipe, we propose a back-of-device word-gesture decoding algorithm which infers the keyboard location from back-of-device gestures, and adjusts the keyboard size to suit the gesture scales; the inferred keyboard is then fed back into the system for decoding. Our user study shows BackSwipe is feasible and a promising input method, especially for command input in the one-hand holding posture: users can enter commands at an average accuracy of 92% with a speed of 5.32 seconds/command. The text entry performance varies across users. The average speed is 9.58 WPM with some users at 18.83 WPM; the average word error rate is 11.04% with some users at 2.85%. Overall, BackSwipe complements the extant smartphone interaction by leveraging the back of the device as a gestural input surface.

CHI '21, May 8–13, 2021, Yokohama, Japan

ACM ISBN 978-1-4503-8096-6/21/05…$15.00
https://doi.org/10.1145/3411764.3445081

CCS CONCEPTS

• **Human-centered computing** → **Gestural input**; **Text input**; *Usability testing*.

KEYWORDS

Text entry; command input; word-gesture shortcuts; touchscreen; smartphones.

ACM Reference Format:
Wenzhe Cui, Suwen Zhu, Zhi Li, Zheer Xu, Xing-Dong Yang, IV Ramakrishnan, and Xiaojun Bi. 2021. BackSwipe: Back-of-device Word-Gesture Interaction on Smartphones. In *CHI Conference on Human Factors in Computing Systems (CHI '21), May 8–13, 2021, Yokohama, Japan.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3411764.3445081

1 INTRODUCTION

One-handed Interaction with a smartphone via the surface on its back (Back-of-device input) has been demonstrated as a promising approach to address some of the well-known problems that are associated with touch input using the thumb, such as "fat finger" or reachability [23, 24]. With the recent development of new smartphone technologies, especially the ones offering a rear display (e.g., Samsung foldable phones), sensing back-of-device input is becoming increasingly practical. However, the bottleneck lies in the software as the existing back-of-device input is largely limited to simple directional strokes or tapping [11, 49, 55]. The full potential of this input modality on a smartphone is yet to be explored.

In this paper, we investigate how word-gesture input (a.k.a ShapeWriting, Swyping, or gesture typing input) [26, 60] can be carried out on an imaginary keyboard to trigger commands or perform text input via the back of a smartphone (Figure 1). Performing word-gesture input on the back of the device has several unique benefits. First, the technique is well suited for one-handed input scenarios as it allows a user to interact with the smartphone with the index finger that is free from gripping the device with one hand (Figure 1). Second, the user can input words or shortcut commands via continuous finger gestures. This is more expressive than simple directional strokes, such as swiping left or right. Third, since user input is carried out on the back of the device, it does not conflict with the system-level gestures performed on the front of the device (e.g., scrolling and swiping).

Despite all these benefits, word-gesture input via the back of the device is challenging because the motion of the index finger is largely restricted by the hand holding the device, causing difficulty for the user to remain accurate in positioning the fingertip on the keyboard. Additionally, the input finger is obscured by the device, which could make the position control more difficult. These factors could result in gestures that are inaccurate and hard to be recognized. A decoder (e.g., $SHARK^2$ decoder [26, 60]) can potentially solve the problem but it requires the knowledge of the location and size of the imaginary keyboard, which is unknown as they differ among people with different hand sizes and holding postures.

To address these challenges, we designed and implemented BackSwipe, a one-handed input technique designed for back-of-device input on a smartphone. BackSwipe infers the location of the imaginary keyboard based on a user's input gestures to create a decoder for back-of-device word-gesture decoding. We have applied this decoder to support both command input and text entry on smartphones.

BackSwipe has the following advantages over extant front-screen one-handed interaction techniques. As a command input method, BackSwipe avoids conflict with existing front-screen gestures. It provides an additional channel for inputting commands and short text which can co-exist with extant front-screen gestures without consuming the affordance of the front screen. As a text entry method, BackSwipe avoids visual interferences with the content displayed on the front screen because there is no need to constantly show a keyboard. It also frees the entire front-screen for interaction with the displayed digital content: Users can freely interact with the digital content on the front in the middle of the text entry. It minimizes the visual and interaction interference on the front screen by providing a dedicated input space on the back for command and text input.

Our user research shows BackSwipe is feasible and promising, especially when a user needs to enter commands or short text in the one-hand holding posture. Our user study shows users can enter commands at an average accuracy of 92% with the speed of 5.32 seconds/command. The text entry performance varies across users. The average speed is 9.58 WPM with some users at 18.83 WPM. The average word error rate is 11.04% with some users at 2.85%.

2 RELATED WORK

Our work is related to the research in word-gesture input systems and back-of-device input.

2.1 Word-Gesture Input for Text and Command Input

Word-gesture typing allows users to enter text using a continuous gesture stroke to traverse the letters on the keyboard. It was first introduced by Zhai and Kristensson [26, 59–61] and has been widely adopted by various commercial keyboards.

Researchers have explored word-gesture typing on a variety of input devices. For example, the bimanual gesture keyboard [6] enabled users to use both hands and multiple strokes to enter one word. i'sFree [67] supported word-gesture input in an eyes-free manner with a touchpad. Chen et al. [8] explored text entry in VR with pressure-sensitive touchscreen devices. These techniques rely on a touchscreen or a touch surface for input, but did not explore other input spaces. Other than using the finger or a stylus, researchers have also extended word-gesture input using head or hand movement. For example, Vulture [40] investigated mid-air word-gesture input by tracking users' hand movement projections on a display. Yu et al. [58] used head rotation for gesture typing in HMDs. Yeo et al. proposed SWiM [56], a tilt-based system to allow single-handed word-gesture input on a smartphone. RotoSwype [19] proposed a ring-based text entry technique that uses the orientation of the ring to support typing. Our work is particularly related to eyes-free gesture typing [67] which also inferred keyboard location for gesture typing. The main difference is that in [67] the keyboard size and x position are known, while for back-of-device input keyboard size, both x and y locations are unknown, which added more complexity for decoding.

In addition to text input, gestures have also been used to trigger commands. Many of the early work in this space, such as marking menus [28–30] and its adaptations, such as hierarchical marking menus [63], wave menus [2], flower menus [3], wavelet menus [16], MarkPad [17], imaginary gestures [20, 21], and M3 gesture menu [64], requires the user to memorize the mapping between a command and its corresponding gesture, thus making it hard for novice users to adopt. Some of the recent research addresses this issue by assigning semantic meanings to the gestures or

using user-defined gesture shortcuts [32, 34–36, 43, 50]. In contrast, word-gesture commands have the benefit that the shape of the gestures directly reflects the name of the corresponding command, thus are easier to memorize and use [1, 12, 27, 65].

Built on the existing word-gesture input techniques and their variants, we explored a new venue of research to investigate how to support word-gesture input on the back of a smartphone (i.e., BackSwipe). Different from the existing word-gesture input methods which require a keyboard to be shown to the user on the front (e.g., SWiM [56]) or touch input to be carried out on the front screen (e.g., one-handed front-screen gesture typing [26, 60]), BackSwipe minimizes the visual and interaction interference on the front screen by providing a dedicated input space on the back for command and text input.

2.2 Back-of-device Interaction on Smartphones

Back-of-device interaction was initially proposed to solve the "fat finger" problem, i.e., direct input on the front of the device may have finger occluding the targets, especially when they are smaller than the finger width [23, 24]. To address this problem, researchers proposed to use additional hardware, such as attaching physical buttons [25, 31, 47] or tactile landmarks [10] on the backside of the device for text entry or triggering applications (e.g., personal calendars). Others explored back-of-device interaction on two-sided touch surfaces [7, 11, 14, 22, 49, 53, 54, 62]. For example, HybridTouch [49] allows the user to interact with a handheld device by simultaneously touching the front display using a stylus and the trackpad on the back using the index finger holding the device. LensGesture [55] detects finger gestures on the back of a smartphone using the device's back-facing camera. XSide [14] allows the user to enter stroke-based passwords via the front or back of the device to protect against shoulder surfing. BackXPress [11] uses pressure sensors on the back of the device to allow the user to switch between different input modes.

Despite all the benefits of back-of-device input (e.g., eliminating the "fat finger" problem, extended input space), a significant trade-off is the occlusion of finger movement. LucidTouch [52] mitigates this problem by using a pseudo-transparency display showing the finger movement on the device's back. NanoTouch [4] is similar except that the device simulates the see-through effect using an image of the finger touching the backside of an ultra-small wearable device.

Existing back-of-device interaction techniques are mostly performed through simple directional strokes (such as swiping up, down, left, or right), or tapping (on physical buttons or different regions of the device). Shimon et al. [48] performed an elicitation study to understand how users map these gestures to smartphone commands. In this work, we extended the gesture space by exploring how to apply word-gesture command input to back-of-device interaction.

3 EXPERIMENT 1: UNDERSTANDING BACK-OF-DEVICE WORD-GESTURE INPUT

We first carried out a study to understand how users performed word-gesture input on the back of a smartphone. Such an understanding would guide us in hldesigning a decoder to support back-of-device word-gesture input.

3.1 Design and Tasks

We conducted a Wizard-of-Oz experiment to collect the participants' gestures on the back of a smartphone. Participants were instructed to hold the phone naturally, imaging an invisible keyboard on the back of the phone which could correctly decode their input. We used a ZTE AXON M as the test device. The device has two 5.2-inch screens, acting as one front and one back screen. In the experiment, the front screen was designed to show prompts, and the back screen was used to collect the gesture input data.

The study adopted a text transcription task. A short phrase was displayed on the front screen and the participant was instructed to draw a word-gesture on the back of the device to enter the phrase. A short line was displayed under the current target word in the phrase and would advance to the next word after the word-gesture for the current word was drawn. The participants were instructed to perform word-gesture input to transcribe text with the index finger on the back screen while holding the phone with the same hand. The gesture traces was displayed on the phone's front screen to provide feedback, as shown in Figure 2. By default, the keyboard layout was not shown on the screen. We adopted such a design because there is evidence showing that many users are able to input text on an imaginary keyboard on the phone [66], remote control [67], or hand-held touchpad [37], given the dominance and users' familiarity with Qwerty layout. In case a user could not recall the location of a particular key, keeping the index finger still for 300ms at the back screen would bring up a Qwerty layout on the front screen. The keyboard would disappear once the user took the index finger off the screen upon the completion of the current word. We instructed the participant to assume that there existed an imaginary keyboard at the back of the screen that would successfully decode the word-gesture she entered.

The testing phrases were selected from a subset of the MacKenzie and Soukoreff phrase set [39, 57]. The same test set was used for all participants. The participants were required to complete 4 blocks in the experiment and could take a short break after completing each block. Each block contains 10 phrases whose orders were randomized. Before the formal study, we instructed the users to complete a 5 minutes warm-up to get familiar with word-gesture input on the back of the device.

3.2 Participants and Apparatus

Ten participants (3 females, all right-handed) participated in the study. The ages of the users were from 25 to 30 (M = 28). Their median familiarity with the Qwerty layout was 4.5 (1: very unfamiliar; 5: very familiar). Their median familiarity with gesture typing was 3. All of the participants were instructed to use their preferred posture to hold the phone during the experiment.

A ZTE AXON M dual-screen foldable mobile phone (Qualcomm MSM8996 Snapdragon processor, Quad-core CPU with a 2×2.15

Figure 2: (a) A participant is entering a word with the index finger on the back of the phone. (b) a screenshot of the front screen. The displayed gesture on the front screen shows the finger trace as if the user is looking through the phone.

GHz Kryo and a 2×1.6 GHz Kryo, Adreno 530 GPU, 4GB RAM, 64GB internal storage, dual 5.2-inch 1080×1920 screen) was used in the study. In the experiment, the front screen was used to display the trial information and other visual guidance. The back screen was used to collect the gesture traces.

In total, the study included: 10 participants × 4 blocks × 10 phrases = 400 trials.

3.3 Results

To understand the positions of the imaginary keys and keyboards, we first inferred the key distribution from the collected gesture traces.

3.3.1 Inferring key positions. Similar to the previous research [67], we used the dynamic time warping (DTW) algorithm [46] to infer the imagined key positions of a word w from its corresponding gesture input g. We first generated a gesture template t for word w. The template was created by connecting the centers of corresponding letters in w on a standard Qwerty layout. The Qwerty layout parameters were obtained from AOSP keyboard for a 1920×1080 resolution Android device. Then we sampled g and t into $N(N = 300)$ equidistant points and applied an optimal match between g and t. In this way, for every letter c in w, we obtained its key center in the sampled template pattern t as t_c, and its corresponding point g_c in the sampled gesture g as the inferred key position. While applying the DTW algorithm to match g with t, we sampled the gesture based on the gesture trace coordinates instead of the timestamps, so the results were unlikely to be affected by the gesturing time between letters.

3.3.2 Distribution of the imaginary keys. Figure 3a shows the distribution of the imaginary key positions inferred from the gestures. The y-direction variance is greater than the variance in the x-direction. The mean standard deviation of the key positions was 4.1 mm and 8.8 mm in x and y-directions. The imaginary keyboard center was mainly located in the upper right part on the back of the phone. We also show the imaginary key distributions of 7 randomly-picked participants in Figure 3b-h. The imaginary keyboard positions differ across users. We also calculated the input speed in the study. The average input speed was 21.9 (SD=6.9) words per minute (WPM).

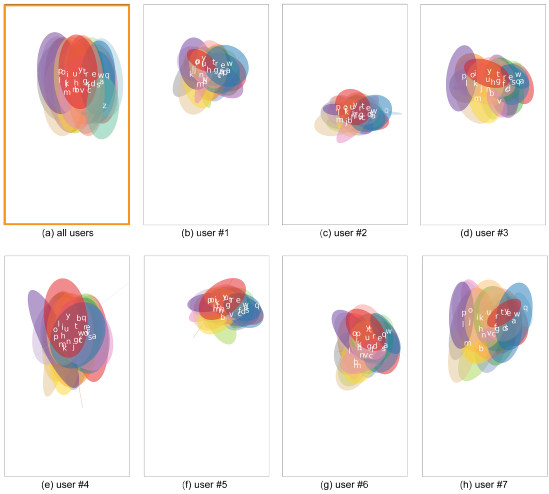


Figure 3: Imaginary keyboard positions and the inferred key distributions (95% confidence ellipses) for back-of-device word-gesture input. The borders illustrate the boundaries of the back screen of the smartphone. (a): the average distribution of all participants. (b) to (h): key distributions obtained from 7 participants randomly picked from all participants.

3.3.3 Discussion. Our investigations led to the following findings, providing a guideline on how to design a decoder for back-of-device word-gesture input.

Users were capable of inputting gestures on the back of the device. Figure 3a shows that the key centers' relative positions on the imaginary keyboard are similar to those on a standard Qwerty layout. Participants were able to recall the Qwerty layout quite accurately, even when they were gesturing on the device's back without the visual of fingers. The average input speed was around 22 WPM, which can be considered the upper limit for back-of-device word-gesture input: the participants were not interrupted by error corrections or candidate selections.

The gesture input behavior on the back of the device was different from regular gesture input. There were significant differences between a back-of-device keyboard layout and the standard Qwerty layout. For instance, different users have varying preferences on where the imaginary keyboard position should be, as shown in Figure 3(b)-(h). The positions of imaginary keyboards differed across users. It is probably because the imaginary keyboard position is easily influenced by the users' postures, which is further affected by their hand size and the coordination of finger movements. We also noticed that users tend to adjust their hand posture when they feel tired or uncomfortable during the study. This indicates that the imaginary keyboard should not be assigned to a fixed position as a regular smartphone keyboard. At the same time, the imaginary keyboard position only undergoes a small change

within the same phrase. Assuming the participants entered every word correctly, we estimated the keyboard position of each gesture. The average keyboard location change in a phrase was 6.5 (SD=2.3) *mm*.

Another interesting finding is that across all the users, the shape of the imaginary keyboard is close to a "square" other than a rectangle: the width to height ratio of a standard Qwerty layout in an AOSP keyboard is $12:5=2.4$, where the imaginary keyboard is $12:9=1.3$

4 BACK-OF-DEVICE WORD-GESTURE DECODING

Based on the findings drawn from the Wizard-of-Oz study, we investigated how to design a decoder for back-of-screen word-gestures. The basic principle of word-gesture decoding [26, 60] is to combine the probability of an intended word estimated from the input gesture (a.k.a spatial probability $c(w)$) with the probability of estimated from the language context (a.k.a language probability $l(w)$) to obtain the overall probability of a word w being the intended input given an input gesture:

$$s(w) = \frac{l(w)c(w)}{\sum_{i \in W} l(i)c(i)}, \quad (1)$$

where W is a lexicon or command set containing i words. We follow the same principle (Equation 1) for decoding back-of-device word-gestures, described as below.

4.1 Spatial Probability

We obtain the spatial probability of a word w by refining the classic $SHARK^2$ decoding algorithm [26, 60], which was originally designed to decode word-gestures drawn on a visible soft keyboard. The core challenge of applying $SHARK^2$ of decoding back-of-device word-gestures is that, in the back-of-device interaction, the size and location of the keyboard are unknown: the imaginary keyboard could be anywhere on the back of the device with any size, depending the holding position. Based on the finding that the keyboard position only undergoes a small change in the same sentence, we created the following method to infer the keyboard location from the input gesture.

We estimate the keyboard size (width and height) based on the data collected in Experiment 1. The study data showed although the keyboard center varies across users, both width and height of the keyboard underwent only small changes across users. The mean width (SD) was 23.9 (4.5) *mm*, and the mean height (SD) was 18.6 (5.5) *mm*. We therefore set the keyboard size to 23.9 ×18.6 *mm*.

Keyboard Position Estimator. We have designed an iterative-updating algorithm to update the center of the imaginary soft keyboard after entering each word. Assuming that the keyboard center for entering the word w_{t-1} is c_{t-1}, the keyboard size is 23.9×18.6 *mm*, and keys are arranged according to a Qwerty layout, we can estimate the center of each key, referred to as $a_{t-1}, b_{t-1}, ..., z_{t-1}$. After a user enters the w_t, we estimate the center of each letter in the word w_t using the DTW algorithm, as described in Section 3.3.1 of Experiment 1. We then calculate the position shift of the corresponding letter, compared it with its previous position for w_{t-1}. The shift of the keyboard center will be the average of all the letter shifts. For example, assuming a user draws a gesture to enter *and*. The iterative-updating algorithm first estimates the new letter centers a_t, n_t, and d_t using the previously described DTW algorithm, and calculates the letter positions shifts $\Delta a = a_t - a_{t-1}$, $\Delta n = n_t - n_{t-1}$, and $\Delta d = d_t - d_{t-1}$. The keyboard position shift is calculated as $\bar{\Delta} = \frac{\Delta a + \Delta n + \Delta d}{3}$.

When entering the first word in a sentence, or a single command, we use only the keyboard size information for decoding since there is no prior knowledge on the keyboard center c. It means that we use only the shape channel of the $SHARK^2$ decoder for decoding, because this channel does not require the information about the keyboard center for calculating the shape similarity between input gestures and word templates. After entering the first word, we then estimate the keyboard center position based on the input gesture, and treat it as c_1. We then apply the iterative-updating algorithm to update the keyboard location after each word, and feed it back into a $SHARK^2$ decoder [26] to decode subsequent gestures.

4.2 Language Probability

For text entry, we used the GPT [44] language model, pretrained by OpenAI. The vocabulary size of the model is 40478. For the first two words in a sentence which have little language context to leverage, we used a 2-gram language model (size: 7 million bigrams) which was trained over the Corpus of Contemporary American English (COCA) [13](2012 to 2017). The Corpus contains over 5 million sentence.

4.3 Decoder Architecture

The workflow of the algorithm is shown in Figure 4. The new component added on top of a regular gesture typing decoder is the keyboard position estimator, as described in the previous section. We view the BackSwipe decoding algorithm as an extension of the eyes-free decoder [67]: extending the algorithm from estimating the y position of the imaginary keyboard only (the keyboard vertical location learner in [67]) to estimating keyboard size, and both x and y position.

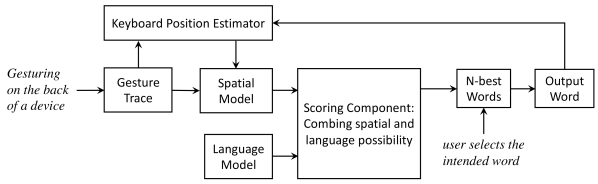


Figure 4: The architecture of BackSwipe's decoder.

4.4 BackSwipe for Text Entry and Command Input

We have applied the decoding algorithm to support both text entry and command input. During text entry, the system will display 4 suggestions, arranged in a 2×2 grid after each gesture input. A user then swipes at the back of the screen in one of the 4 directions to select the corresponding suggestion. In case the user cannot recall the key position on a Qwerty layout, she may hold the finger still on

the back of the device for 0.3 seconds to invoke a Qwerty keyboard displayed on the front screen.

We have also applied BackSwipe to support command input, by replacing the lexicon with a set of commands. A user triggers a command by back-swiping (i.e., drawing a word-gesture at the back of the device) the corresponding command label. By default, the recognized command will be executed upon the input finger lifts off the back surface. If the recognition scores of the top two candidates are close enough (i.e., the score difference is less than 5% of the average score), the system will display 4 command candidates, and the user swipes on the back screen in the corresponding direction to trigger the command. Similar to text entry, holding the input finger still on the back for 0.3 seconds will trigger the display of a Qwerty layout on the front screen to help the user recall key positions.

We have implemented a BackSwipe decoder as described in this section on a ZTE AXON M dual-screen foldable phone which ran Android 7.1.2. The source code can be found at Bitbucket.[1] Next, we carried out two experiments to evaluate BackSwipe, for command input and text entry separately.

5 EXPERIMENT 2: EVALUATING BACKSWIPE FOR COMMAND INPUT

We carried out Experiment 2 to evaluate BackSwipe for command input.

5.1 Design

This experiment is a command activation task. The participants were required to complete multiple command activation trials using BackSwipe, among a set of 30 commands. The size of the command set was close or bigger than the size of commonly used command sets in a mobile application (e.g., a Chrome web browser supports 13 commands in its first-layer menu).

Before the experiment, the participants were shown 30 target commands and their corresponding stimulus icons on a sheet. The users were required to memorize at least 90% of the commands. Each command could be triggered by one of the two words. For example, to trigger the "clock" icon, the users could enter either "clock" or "timer". The words were created from [15] by judging whether they reflected the meaning of the icon. The commands and corresponding words for triggering them are shown in Appendix (Table 2). Since this experiment was not a text composition or transcription task, we replaced the lexicon in the decoder with a set of 60 words (2 words per command).

5.2 Procedure

At the start of each trial, a stimulus icon was displayed at the upper part of the front screen. The participants were required to input the command by BackSwipe. After each gesture input, the decoder would analyze the input confidence and decide whether to provide command candidates following the policy described in Section 4.4, as shown in Figure 5. If candidates were provided, participants needed to select the intended command by swiping to the corresponding direction on the device's back. If not, the first candidate command from the decoder was used as the default command output. A trial was completed if the input command was correct, or the participants made three consecutive failed attempts. Before the formal test, the participants were instructed to get familiar with the test procedure and BackSwipe in a warm-up session of 5 minutes. The commands in the warm-up session would not appear in the formal test.

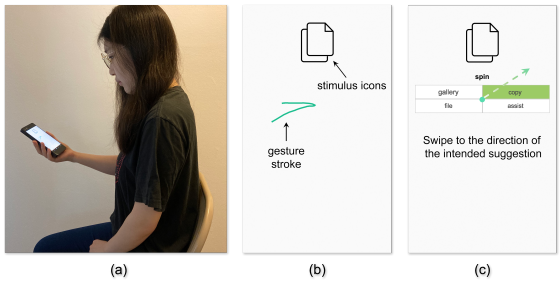


Figure 5: (a) A user is inputting the command "copy" with BackSwipe. (b) The stimulus icon is displayed on the front screen. (c) The decoder shows the command "spin" with four more candidate commands. Since "spin" is not the intended command, the participant swipes her finger to the direction of the intended word "copy" as if the finger is located at the center of the suggestion grid (i.e., top-right direction). In other words, drawing a gesture towards top-right corner would select the "copy" command

We randomly selected 12 out of 30 commands from the command set. Based on previous research, the command input patterns of launching applications [41] or selecting menu items [9, 33] usually follow certain distributions (e.g. Zipfian distribution). We generated the command occurrence frequency according to Zipfian distribution. The occurrences for the 12 commands was (7, 5, 4, 4, 2, 2, 1, 1, 1, 1, 1, 1). We counterbalanced the test commands to frequencies across all participants, ensuring that each command maps to each frequency the same number of times. The same set of commands was used across all participants. The order of the commands was randomized for each participant. The experiment was divided into 5 blocks, each consisting of 6 trials.

5.3 Participants and Apparatus

12 participants (3 females, all right-handed) from 25 to 29 years old took part in the study. The self-reported median familiarity (1: very not familiar; 5 very familiar) with word-gesture input and Qwerty layout were 3 and 4, respectively. We instructed the participants to complete the experiment with their preferred hand postures. The same ZTE AXON M device was used.

In total, the study included: 12 participants × 5 blocks × 6 phrases = 360 trials.

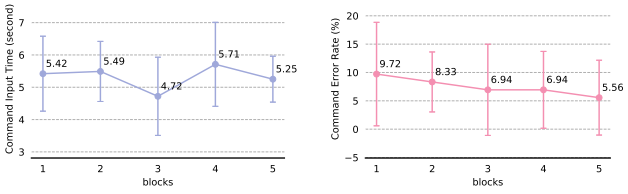
5.4 Results

Command Input Time. This metric indicates how fast a user could trigger a command on the back of the device. It is measured as the elapsed time from the moment a participant starts drawing a gesture to the end of the trial. The average command input time was

[1] https://bitbucket.org/wenzhecui/backswipe

5.32 (SD=1.28) seconds. To understand the performance of the participants as they progressed in the experiment, we show the mean (95% CI) command input time across the five blocks in Figure 6a. A repeated-measures ANOVA did not show a significant main effect of blocks on the command input time ($F_{4,44} = 0.86, p = 0.49$).

Command Input Error Rate. The command input error rate was the ratio of failed trials over the total number of trials. The average command input error rate among all participants was 7.50% (SD=6.05%). The mean (95% CI) command input error rate across the five blocks is shown in Figure 6b. A repeated-measures ANOVA did not show a significant main effect of blocks on error rate ($F_{4,44} = 0.23, p = 0.92$).



(a) Command input time **(b) Command input error rate**

Figure 6: Average (95% CI) of command input time (in seconds) and command error rate (%) by test blocks.

Suggestion Usage. This metric indicates how useful the command suggestions were in the experiment. It was calculated as the ratio of successful commands triggered using suggestions over the total number of successful trials. The average suggestion usage was 29.8% (SD=11.0%), indicating that for over 70% of successful command trials users did not need suggestions.

Reference Keyboard Usage This metric measures how often the reference keyboard was invoked on the front screen during the experiment to guide the user input. It was calculated as the number of times the keyboard showed up on the screen over the total number of input gestures. The average reference keyboard usage was 0.43 (SD=0.05) times, indicating that 43% of times the reference keyboard was invoked.

Subjective Feedback. At the end of the experiment, participants were required to provide a continuous numerical rating (1: least demanding, 10: most demanding) on the experiment's mental and physical demands. Mental demand described how much mental effort was required to complete the study. Physical demand described how much physical effort was required. The average mental demand was 4.9 (SD=2.3). The average physical demand was 5.7 (SD=2.3). The participants were also asked to rate the BackSwipe according to their impression on a scale of 1 to 5 (1: dislike, 5: like). The average rating was 3.8 (SD=0.9).

5.5 Discussion

Our experiment led to the following findings.

First, BackSwipe is a promising method to input gesture command. The results showed that users could successfully enter commands at an average error rate of 7.5%, at the speed of 5.32 seconds per command. Four users (1/3 of the participants) entered commands at a speed faster than 4 seconds/command. It shows that BackSwipe can help users to conveniently enter command shortcuts in a one hand holding posture.

Second, suggestions are necessary for triggering commands with similar shapes, such as "cut" and "copy". Entering these commands requires high input accuracy, which might lead to more errors. Participants also tend to make more errors in complicated gestures. For example, to input the command "delete", users need to trace their fingers back and force around the letter "e", and thus would make more mistakes. With the help of command suggestions, users could finish the task more easily are relaxed.

6 EXPERIMENT 3: EVALUATING BACKSWIPE FOR TEXT ENTRY

After evaluating BackSwipe for command input, we carried out Experiment 3 to evaluate it for text entry.

6.1 Design and Tasks

This study was a text transcription task. Similar to the design in Experiment 1, the participants were instructed to transcribe the phrases shown on the front screen with back-of-device gesture input by BackSwipe. We did not include other back-of-device input methods or the front-screen one-handed text entry method as baselines, because (1) the existing back-of-device input technologies are mostly simple-gesture or tap-based, which do not support text entry, and (2) the existing one-handed text entry methods require constantly displaying a soft keyboard (e.g., [56]), or touch input on the front screen (e.g., [67]). BackSwipe was created to avoid visual and interaction interference with the front screen content, which worked work under different constraints than other front-screen one-handed text entry methods.

The testing phrases were also selected from a subset of the MacKenzie and Soukoreff phrase set [39, 57]. All the phrases in this study were different from those in Experiment 1. The participants were required to transcribe the same set of 15 phrases, divided into 5 blocks. Each block contained 3 phrases. The order of the 15 phrases was randomized. The participants were allowed to take a break after completing each block. Same as regular soft keyboards, for each back-of-device gesture input, BackSwipe generates the top candidate word on the input area and 4 additional words as suggestions. If the top candidate was not the intended word, the participant could swipe towards a direction to select the corresponding suggestion, as shown in Figure 7. If the top candidate was correct, the user tapped the back of the screen to commit it. If all the decoding results were incorrect, the participant could press the "volume down" key on the edge of the device as backspace. Before the formal test, subjects performed a warm-up session for around 5 minutes for them to get familiar with BackSwipe. The phrases in the warm-up session were different from those in the formal test.

6.2 Participants and Apparatus

12 subjects (4 females, all right-handed) from 25 to 29 years old were recruited in the study. The self-reported median familiarity (1: very not familiar; 5 very familiar) with word-gesture input and Qwerty layout was 3 and 4.5, respectively. We instructed the users to completed the study using their preferred hand postures. The same ZTE AXON M device was used.

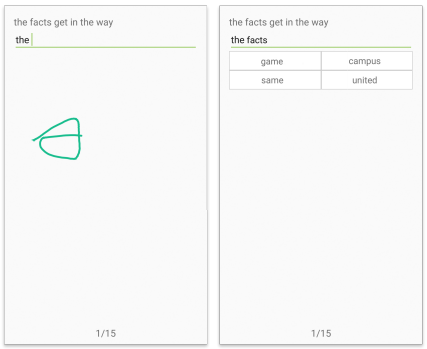


Figure 7: (a) A user is gesturing the word "facts" with BackSwipe. (b) When the top candidate is not the intended word, the user could swipe her finger to the direction of the word "facts" from the center of the suggestion grid to select it.

In total, the study included: 12 participants × 5 blocks × 3 phrases = 180 trials.

6.3 Results

Input Speed. We use word per minute (WPM) to measure the input speed. This measure indicates how fast a user could input text with BackSwipe. The calculation followed the method proposed in the previous work [38]:

$$WPM = \frac{|S-1|}{T} \times \frac{1}{5}, \tag{2}$$

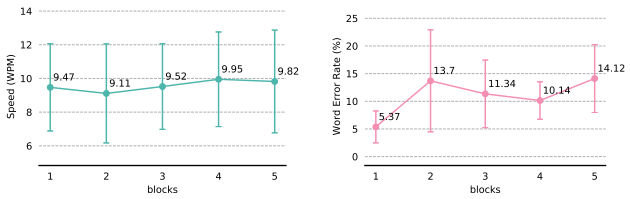
where S is the length of the transcribed text in character, including spaces, and T is the elapsed time in minutes from the start of the first gesture stroke to finishing the last word in the phrase. The average input speed across all participants was 9.58 (SD=4.10) WPM. To understand how the participants performed over time, we plotted the mean (95% CI) input speed across the five blocks in Figure 8a. A repeated-measures ANOVA did not show a significant main effect of block on input speed ($F_{4,44} = 0.25, p = 0.91$).

Word Error Rate. Gesture input on the back of the device was on word level. We measured the error rate using word error rate [5, 66]:

$$r = \frac{WD(E,T)}{|T|} \times 100\%, \tag{3}$$

where $WD(E,T)$ is the word edit distance between the entered (transcribed) phrase E and the target phrase T, and $|T|$ is the number of words in T. The word edit distance is the minimum number of basic word-level operations needed to transform the transcribed phrase into the target phrase. The word-level operations are insertion, replacement and deletion. The average word error rate among all participants was 11.04% (SD=4.87%). Figure 8b shows the mean (95% CI) word error rate across the five blocks. A repeated-measures ANOVA did not show a significant effect of block on word error rate ($F_{4,44} = 1.70, p = 0.17$).

Suggestion Usage. We analyzed the average number of suggestion usage for each input gesture. For every gesture, if the participant used the top candidate as the output word, no (0) suggestion was used. If the participant picked one of the four suggestion words, 1 suggestion is used. The average number of suggestion used among all participants was 0.36 (SD=0.07).



(a) Input speed **(b) Word error rate**

Figure 8: Average (95% CI) of text input speed (WPM) and word error rate (%) by block.

Reference Keyboard Usage We also calculated the reference keyboard usage for the text transcription task. It was calculated as the number of times the keyboard showed up on the screen over the total number of input gestures. The reference keyboard showed up an average of 0.60 (SD=0.12) times for each back-of-device gesture.

Subjective Feedback. At the end of the experiment, participants were required to provide a numerical rating (1: least demanding, 10: most demanding) on the experiment's mental and physical demands. The average mental demand and physical demand rating were 5.6 (SD=2.3) and 6.8 (SD=2.0), respectively. The participants also rated BackSwipe for text input on a scale of 1 to 5 (1: dislike, 5: like). The average score was 3.1 (SD=0.8).

6.4 Discussion

Our study reveals that it is feasible to input text with BackSwipe, with the average speed of 9.58 WPM. However, due to the limited flexibility, users cannot enter text as fast as on the front screen. Some users also commented that using BackSwipe over a long period of time caused fatigue. Based on the study results, it would be more practical to use BackSwipe for short or quick text entry such as replying to a short message. It is probably not appropriate for extended text entry such as writing emails. Users' ability of using BackSwipe also differs. The input speeds ranged from 5.87 to 18.83 WPM among 12 participants. The performance varied across users.

Table 1 shows the input speed and accuracy of BackSwipe in comparison with other input methods. As shown, the performance of BackSwipe for inputting short words (e.g., command names) is comparable with other front-screen keyboard-visible input methods, while its performance decreases for inputting phrase level-text. It further suggests that BackSwipe is more suitable for short text input (e.g., command names).

The promising performance of BackSwipe (Table 1) and its unique affordance of being a back-of-device, gestural input method suggest that it has pros and cons compared with simple-gesture or menu based back-of-device input methods. BackSwipe is likely to be faster than other back-of-device input methods when triggering commands among a large number of candidates, because BackSwipe is essentially a recall-based command input method: a user can directly trigger a command by gesturing the command name. On the contrary, simple gestures or menu-based command input methods

Table 1: Input speed and accuracy of BackSwipe and other input methods from literature.

Affordance	Input Method	Input Speed (WPM)	Character Error Rate (%)
Front screen keyboard visible	Gesture Keyboards [45]	25 - 31	1.0% - 3.6%
	SWiM [56]	15	0.9%
	ZoomBoard [42]	8 - 9	0.7%
	WatchWriter [18]	24	3.7%
	VelociWatch [51]	17	3.0%
BackScreen keyboard invisible	BackSwipe (command)	18	7.5%
	BackSwipe (phrases)	10	10.0%

may involve extensive searching and navigation operations before a command can be selected (e.g., in a hierarchical menu). On the other hand, simple gesture or menu based methods might have an advantage over BackSwipe if the number of available commands is small (e.g., 2 or 3). BackSwipe increases the expressiveness of back-of-device interaction by enabling short text and word-gesture command input, while extant simple-gesture based back-of-device input methods have little support for text input. Additionally, as it is difficult to sustain long and intensive input at the back of the device, BackSwipe is recommended for short text and word-gesture command input, but not for intensive text input. Given its pros and cons, we consider BackSwipe to be a complement to the existing front-screen and back-of-device interaction methods. For example, to quickly switch to the camera application, BackSwipe allows a user to activate it by inputting the word-gesture "camera" on the back of the device, without returning to the home screen and selecting the camera icon. A user may BackSwipe a short message while she is watching a video or attending an online meeting via the smartphone. Using BackSwipe allows the user to input short text without interrupting the ongoing front-screen interaction.

7 CONCLUSIONS

We have designed and implemented BackSwipe, a back-of-device command and text input technique. It supports a user to hold a smartphone with one hand and use the index finger of the same hand to draw a word-gesture anywhere at the back of the device to enter a command and text. The key for supporting BackSwipe is the back-of-device word-gesture decoding algorithm, which infers the keyboard location from back-screen gestures, and adjusts the keyboard size to suit the gesture scales; the inferred keyboard is then fed back into the system for decoding. Based on this decoding algorithm, we have implemented BackSwipe to support command and text input. Our evaluation shows that BackSwipe is promising for supporting command input: users can enter commands at an average accuracy of 92% with a speed of 5.32 seconds/command. The text entry performance varies across users. The average speed is 9.58 WPM with some users at 18.8 WPM; the average word error rate is 11.04% with some users at 2.85%. Overall, our research contributes knowledge on decoding back-of-device gesture input and shows that BackSwipe is a feasible and promising input method for one-handed interaction on smartphones, and serves as a complement to the existing front-screen and back-of-device interaction methods.

ACKNOWLEDGMENTS

We thank anonymous reviewers for their insightful comments, and our user study participants. This work was supported by NSF awards 1815514, 1805076, 1936027, NIH R01EY030085, R01HD097188 ALS Association grant 20-MALS-538 and NIDILRR award: 90IFO117-01-00. This work was done as part of the Ph.D. dissertation of Wenzhe Cui, a Stony Brook Ph.D. student supervised by Dr. Xiaojun Bi.

REFERENCES

[1] Jessalyn Alvina, Carla F. Griggio, Xiaojun Bi, and Wendy E. Mackay. 2017. CommandBoard: Creating a General-Purpose Command Gesture Input Space for Soft Keyboard. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Québec City, QC, Canada) *(UIST '17)*. ACM, New York, NY, USA, 17–28. https://doi.org/10.1145/3126594.3126639

[2] Gilles Bailly, Eric Lecolinet, and Laurence Nigay. 2007. Wave Menus: Improving the Novice Mode of Hierarchical Marking Menus. In *Proceedings of the 11th IFIP TC 13 International Conference on Human-computer Interaction* (Rio de Janeiro, Brazil) *(INTERACT'07)*. Springer-Verlag, Berlin, Heidelberg, 475–488. http://dl.acm.org/citation.cfm?id=1776994.1777053

[3] Gilles Bailly, Eric Lecolinet, and Laurence Nigay. 2008. Flower Menus: A New Type of Marking Menu with Large Menu Breadth, Within Groups and Efficient Expert Mode Memorization. In *Proceedings of the Working Conference on Advanced Visual Interfaces* (Napoli, Italy) *(AVI '08)*. ACM, New York, NY, USA, 15–22. https://doi.org/10.1145/1385569.1385575

[4] Patrick Baudisch and Gerry Chu. 2009. Back-of-Device Interaction Allows Creating Very Small Touch Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Boston, MA, USA) *(CHI '09)*. Association for Computing Machinery, New York, NY, USA, 1923–1932. https://doi.org/10.1145/1518701.1518995

[5] Xiaojun Bi, Shiri Azenkot, Kurt Partridge, and Shumin Zhai. 2013. Octopus: Evaluating Touchscreen Keyboard Correction and Recognition Algorithms via. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. ACM, New York, NY, USA, 543–552. https://doi.org/10.1145/2470654.2470732

[6] Xiaojun Bi, Ciprian Chelba, Tom Ouyang, Kurt Partridge, and Shumin Zhai. 2012. Bimanual Gesture Keyboard. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology* (Cambridge, Massachusetts, USA) *(UIST '12)*. ACM, New York, NY, USA, 137–146. https://doi.org/10.1145/2380116.2380136

[7] Daniel Buschek, Oliver Schoenleben, and Antti Oulasvirta. 2014. Improving Accuracy in Back-of-Device Multitouch Typing: A Clustering-Based Approach to Keyboard Updating. In *Proceedings of the 19th International Conference on Intelligent User Interfaces* (Haifa, Israel) *(IUI '14)*. Association for Computing Machinery, New York, NY, USA, 57–66. https://doi.org/10.1145/2557500.2557501

[8] Sibo Chen, Junce Wang, Santiago Guerra, Neha Mittal, and Soravis Prakkamakul. 2019. Exploring Word-Gesture Text Entry Techniques in Virtual Reality. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI EA '19)*. Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/3290607.3312762

[9] Andy Cockburn, Carl Gutwin, and Saul Greenberg. 2007. A Predictive Model of Menu Performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '07)*. ACM, New York, NY, USA, 627–636. https://doi.org/10.1145/1240624.1240723

[10] Christian Corsten, Christian Cherek, Thorsten Karrer, and Jan Borchers. 2015. HaptiCase: Back-of-Device Tactile Landmarks for Eyes-Free Absolute Indirect Touch. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in*

Computing Systems (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 2171–2180. https://doi.org/10.1145/2702123.2702277

[11] Christian Corsten, Bjoern Daehlmann, Simon Voelker, and Jan Borchers. 2017. BackXPress: Using Back-of-Device Finger Pressure to Augment Touchscreen Input on Smartphones. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 4654–4666. https://doi.org/10.1145/3025453.3025565

[12] Wenzhe Cui, Jingjie Zheng, Blaine Lewis, Daniel Vogel, and Xiaojun Bi. 2019. HotStrokes: Word-Gesture Shortcuts on a Trackpad. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. ACM, New York, NY, USA, Article 165, 13 pages. https://doi.org/10.1145/3290605.3300395

[13] Mark Davies. 2018. The corpus of contemporary American English: 1990-present.

[14] Alexander De Luca, Marian Harbach, Emanuel von Zezschwitz, Max-Emanuel Maurer, Bernhard Ewald Slawik, Heinrich Hussmann, and Matthew Smith. 2014. Now You See Me, Now You Don't: Protecting Smartphone Authentication from Shoulder Surfers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 2937–2946. https://doi.org/10.1145/2556288.2557097

[15] Dictionary.com. 2020. Thesaurus.com | Synonyms and Antonyms of Words. https://www.thesaurus.com/. https://www.thesaurus.com/ [Online; accessed 13-August-2020].

[16] Jeremie Francone, Gilles Bailly, Laurence Nigay, and Eric Lecolinet. 2009. Wavelet Menus: A Stacking Metaphor for Adapting Marking Menus to Mobile Devices. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services* (Bonn, Germany) *(MobileHCI '09)*. ACM, New York, NY, USA, Article 49, 4 pages. https://doi.org/10.1145/1613858.1613919

[17] Bruno Fruchard, Eric Lecolinet, and Olivier Chapuis. 2017. MarkPad: Augmenting Touchpads for Command Selection. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. ACM, New York, NY, USA, 5630–5642. https://doi.org/10.1145/3025453.3025486

[18] Mitchell Gordon, Tom Ouyang, and Shumin Zhai. 2016. WatchWriter: Tap and Gesture Typing on a Smartwatch Miniature Keyboard with Statistical Decoding. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 3817–3821. https://doi.org/10.1145/2858036.2858242

[19] Aakar Gupta, Cheng Ji, Hui-Shyong Yeo, Aaron Quigley, and Daniel Vogel. 2019. RotoSwype: Word-Gesture Typing Using a Ring. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300244

[20] Sean Gustafson. 2012. Imaginary interfaces: touchscreen-like interaction without the screen. In *CHI'12 Extended Abstracts on Human Factors in Computing Systems*. 927–930.

[21] Sean Gustafson, Christian Holz, and Patrick Baudisch. 2011. Imaginary phone: learning imaginary interfaces by transferring spatial memory from a familiar device. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. 283–292.

[22] Hiroyuki Hakoda, Yoshitomo Fukatsu, Buntarou Shizuki, and Jiro Tanaka. 2015. Back-of-Device Interaction Based on the Range of Motion of the Index Finger. In *Proceedings of the Annual Meeting of the Australian Special Interest Group for Computer Human Interaction* (Parkville, VIC, Australia) *(OzCHI '15)*. Association for Computing Machinery, New York, NY, USA, 202–206. https://doi.org/10.1145/2838739.2838812

[23] Christian Holz and Patrick Baudisch. 2010. The Generalized Perceived Input Point Model and How to Double Touch Accuracy by Extracting Fingerprints. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Atlanta, Georgia, USA) *(CHI '10)*. Association for Computing Machinery, New York, NY, USA, 581–590. https://doi.org/10.1145/1753326.1753413

[24] Christian Holz and Patrick Baudisch. 2011. Understanding Touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 2501–2510. https://doi.org/10.1145/1978942.1979308

[25] Hwan Kim, Yea-kyung Row, and Geehyuk Lee. 2012. Back Keyboard: A Physical Keyboard on Backside of Mobile Phone Using Qwerty. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems* (Austin, Texas, USA) *(CHI EA '12)*. Association for Computing Machinery, New York, NY, USA, 1583–1588. https://doi.org/10.1145/2212776.2223676

[26] Per-Ola Kristensson and Shumin Zhai. 2004. SHARK2: A Large Vocabulary Shorthand Writing System for Pen-based Computers. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology* (Santa Fe, NM, USA) *(UIST '04)*. ACM, New York, NY, USA, 43–52. https://doi.org/10.1145/1029632.1029640

[27] Per Ola Kristensson and Shumin Zhai. 2007. Command Strokes with and Without Preview: Using Pen Gestures on Keyboard for Command Selection. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '07)*. ACM, New York, NY, USA, 1137–1146. https://doi.org/10.1145/1240624.1240797

[28] 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* (Boston, Massachusetts, USA) *(CHI '94)*. ACM, New York, NY, USA, 258–264. https://doi.org/10.1145/191666.191759

[29] Gordon Paul Kurtenbach. 1993. *The Design and Evaluation of Marking Menus*. Ph.D. Dissertation. University of Toronto, Toronto, Ont., Canada, Canada. UMI Order No. GAXNN-82896.

[30] Gordon P. Kurtenbach, Abigail J. Sellen, and William A. S. Buxton. 1993. An Empirical Evaluation of Some Articulatory and Cognitive Aspects of Marking Menus. *Hum.-Comput. Interact.* 8, 1 (March 1993), 1–23. https://doi.org/10.1207/s15327051hci0801_1

[31] Kevin A. Li, Patrick Baudisch, and Ken Hinckley. 2008. Blindsight: Eyes-Free Access to Mobile Phones. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Florence, Italy) *(CHI '08)*. Association for Computing Machinery, New York, NY, USA, 1389–1398. https://doi.org/10.1145/1357054.1357273

[32] Yang Li. 2010. Gesture Search: A Tool for Fast Mobile Data Access. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology* (New York, New York, USA) *(UIST '10)*. ACM, New York, NY, USA, 87–96. https://doi.org/10.1145/1866029.1866044

[33] Wanyu Liu, Gilles Bailly, and Andrew Howes. 2017. Effects of Frequency Distribution on Linear Menu Performance. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. ACM, New York, NY, USA, 1307–1312. https://doi.org/10.1145/3025453.3025707

[34] Hao Lü and Yang Li. 2011. Gesture Avatar: A Technique for Operating Mobile User Interfaces Using Gestures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. ACM, New York, NY, USA, 207–216. https://doi.org/10.1145/1978942.1978972

[35] Hao Lü and Yang Li. 2013. Gesture Studio: Authoring Multi-touch Interactions Through Demonstration and Declaration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. ACM, New York, NY, USA, 257–266. https://doi.org/10.1145/2470654.2470690

[36] Hao Lü and Yang Li. 2015. Gesture On: Enabling Always-On Touch Gestures for Fast Mobile Access from the Device Standby Mode. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. ACM, New York, NY, USA, 3355–3364. https://doi.org/10.1145/2702123.2702610

[37] Yiqin Lu, Chun Yu, Xin Yi, Yuanchun Shi, and Shengdong Zhao. 2017. BlindType: Eyes-Free Text Entry on Handheld Touchpad by Leveraging Thumb's Muscle Memory. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 2, Article 18 (June 2017), 24 pages. https://doi.org/10.1145/3090083

[38] I. Scott MacKenzie. 2015. *A Note on Calculating Text Entry Speed.*

[39] I. Scott MacKenzie and R. William Soukoreff. 2003. Phrase Sets for Evaluating Text Entry Techniques. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems* (Ft. Lauderdale, Florida, USA) *(CHI EA '03)*. ACM, New York, NY, USA, 754–755. https://doi.org/10.1145/765891.765971

[40] Anders Markussen, Mikkel Rønne Jakobsen, and Kasper Hornbæk. 2014. Vulture: A Mid-air Word-gesture Keyboard. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. ACM, New York, NY, USA, 1073–1082. https://doi.org/10.1145/2556288.2556964

[41] Alistair Morrison, Xiaoyu Xiong, Matthew Higgs, Marek Bell, and Matthew Chalmers. 2018. A Large-Scale Study of iPhone App Launch Behaviour. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. ACM, New York, NY, USA, Article 344, 13 pages. https://doi.org/10.1145/3173574.3173918

[42] Stephen Oney, Chris Harrison, Amy Ogan, and Jason Wiese. 2013. ZoomBoard: A Diminutive Qwerty Soft Keyboard Using Iterative Zooming for Ultra-Small Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 2799–2802. https://doi.org/10.1145/2470654.2481387

[43] Benjamin Poppinga, Alireza Sahami Shirazi, Niels Henze, Wilko Heuten, and Susanne Boll. 2014. Understanding Shortcut Gestures on Mobile Touch Devices. In *Proceedings of the 16th International Conference on Human-Computer Interaction with Mobile Devices & Services* (Toronto, ON, Canada) *(MobileHCI '14)*. Association for Computing Machinery, New York, NY, USA, 173–182. https://doi.org/10.1145/2628363.2628378

[44] A. Radford. 2018. Improving Language Understanding by Generative Pre-Training.

[45] Shyam Reyal, Shumin Zhai, and Per Ola Kristensson. 2015. Performance and User Experience of Touchscreen and Gesture Keyboards in a Lab Setting and in the Wild. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 679–688. https://doi.org/10.1145/2702123.2702597

[46] Hiroaki Sakoe and Seibi Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing* 26, 1 (February 1978), 43–49. https://doi.org/10.1109/TASSP.1978.1163055

[47] James Scott, Shahram Izadi, Leila Sadat Rezai, Dominika Ruszkowski, Xiaojun Bi, and Ravin Balakrishnan. 2010. RearType: Text Entry Using Keys on the Back of a Device. In *Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services* (Lisbon, Portugal) *(MobileHCI '10)*. Association for Computing Machinery, New York, NY, USA, 171–180. https://doi.org/10.1145/1851600.1851630

[48] Shaikh Shawon Arefin Shimon, Sarah Morrison-Smith, Noah John, Ghazal Fahimi, and Jaime Ruiz. 2015. Exploring User-Defined Back-Of-Device Gestures for Mobile Devices. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services* (Copenhagen, Denmark) *(MobileHCI '15)*. Association for Computing Machinery, New York, NY, USA, 227–232. https://doi.org/10.1145/2785830.2785890

[49] Masanori Sugimoto and Keiichi Hiroki. 2006. HybridTouch: An Intuitive Manipulation Technique for PDAs Using Their Front and Rear Surfaces. In *Proceedings of the 8th Conference on Human-Computer Interaction with Mobile Devices and Services* (Helsinki, Finland) *(MobileHCI '06)*. Association for Computing Machinery, New York, NY, USA, 137–140. https://doi.org/10.1145/1152215.1152243

[50] Radu-Daniel Vatavu and Ovidiu-Ciprian Ungurean. 2019. Stroke-Gesture Input for People with Motor Impairments: Empirical Results & Research Roadmap. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3290605.3300445

[51] Keith Vertanen, Dylan Gaines, Crystal Fletcher, Alex M. Stanage, Robbie Watling, and Per Ola Kristensson. 2019. VelociWatch: Designing and Evaluating a Virtual Keyboard for the Input of Challenging Text. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3290605.3300821

[52] Daniel Wigdor, Clifton Forlines, Patrick Baudisch, John Barnwell, and Chia Shen. 2007. Lucid Touch: A See-through Mobile Device. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology* (Newport, Rhode Island, USA) *(UIST '07)*. Association for Computing Machinery, New York, NY, USA, 269–278. https://doi.org/10.1145/1294211.1294259

[53] Daniel Wigdor, Darren Leigh, Clifton Forlines, Samuel Shipman, John Barnwell, Ravin Balakrishnan, and Chia Shen. 2006. Under the Table Interaction. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology* (Montreux, Switzerland) *(UIST '06)*. Association for Computing Machinery, New York, NY, USA, 259–268. https://doi.org/10.1145/1166253.1166294

[54] Jacob O. Wobbrock, Brad A. Myers, and Htet Htet Aung. 2008. The Performance of Hand Postures in Front- and Back-of-Device Interaction for Mobile Computing. *Int. J. Hum.-Comput. Stud.* 66, 12 (Dec. 2008), 857–875. https://doi.org/10.1016/j.ijhcs.2008.03.004

[55] Xiang Xiao, Teng Han, and Jingtao Wang. 2013. LensGesture: Augmenting Mobile Interactions with Back-of-Device Finger Gestures. In *Proceedings of the 15th ACM on International Conference on Multimodal Interaction* (Sydney, Australia) *(ICMI '13)*. Association for Computing Machinery, New York, NY, USA, 287–294. https://doi.org/10.1145/2522848.2522850

[56] Hui-Shyong Yeo, Xiao-Shen Phang, Steven J. Castellucci, Per Ola Kristensson, and Aaron Quigley. 2017. Investigating Tilt-based Gesture Keyboard Entry for Single-Handed Text Entry on Large Devices. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. ACM, New York, NY, USA, 4194–4202. https://doi.org/10.1145/3025453.3025520

[57] Xin Yi, Chun Yu, Weinan Shi, Xiaojun Bi, and Yuanchun Shi. 2017. Word Clarity As a Metric in Sampling Keyboard Test Sets. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. ACM, New York, NY, USA, 4216–4228. https://doi.org/10.1145/3025453.3025701

[58] Chun Yu, Yizheng Gu, Zhican Yang, Xin Yi, Hengliang Luo, and Yuanchun Shi. 2017. Tap, Dwell or Gesture?: Exploring Head-Based Text Entry Techniques for HMDs. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. ACM, New York, NY, USA, 4479–4488. https://doi.org/10.1145/3025453.3025964

[59] Shumin Zhai and Per-Ola Kristensson. 2003. Shorthand Writing on Stylus Keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Ft. Lauderdale, Florida, USA) *(CHI '03)*. ACM, New York, NY, USA, 97–104. https://doi.org/10.1145/642611.642630

[60] Shumin Zhai and Per Ola Kristensson. 2012. The Word-gesture Keyboard: Reimagining Keyboard Interaction. *Commun. ACM* 55, 9 (Sept. 2012), 91–101. https://doi.org/10.1145/2330667.2330689

[61] Shumin Zhai, Per Ola Kristensson, Pengjun Gong, Michael Greiner, Shilei Allen Peng, Liang Mico Liu, and Anthony Dunnigan. 2009. Shapewriter on the Iphone: From the Laboratory to the Real World. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems* (Boston, MA, USA) *(CHI EA '09)*. ACM, New York, NY, USA, 2667–2670. https://doi.org/10.1145/1520340.1520380

[62] Cheng Zhang, Anhong Guo, Dingtian Zhang, Caleb Southern, Rosa Arriaga, and Gregory Abowd. 2015. BeyondTouch: Extending the Input Language with Built-in Sensors on Commodity Smartphones. In *Proceedings of the 20th International Conference on Intelligent User Interfaces* (Atlanta, Georgia, USA) *(IUI '15)*. Association for Computing Machinery, New York, NY, USA, 67–77. https://doi.org/10.1145/2678025.2701374

[63] Shengdong Zhao and Ravin Balakrishnan. 2004. Simple vs. Compound Mark Hierarchical Marking Menus. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology* (Santa Fe, NM, USA) *(UIST '04)*. ACM, New York, NY, USA, 33–42. https://doi.org/10.1145/1029632.1029639

[64] Jingjie Zheng, Xiaojun Bi, Kun Li, Yang Li, and Shumin Zhai. 2018. M3 Gesture Menu: Design and Experimental Analyses of Marking Menus for Touchscreen Mobile Interaction. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. ACM, New York, NY, USA, Article 249, 14 pages. https://doi.org/10.1145/3173574.3173823

[65] Suwen Zhu, Yoonsang Kim, Jingjie Zheng, Jennifer Yi Luo, Ryan Qin, Liuping Wang, Xiangmin Fan, Feng Tian, and Xiaojun Bi. 2020. Using Bayes' Theorem for Command Input: Principle, Models, and Applications. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–15. https://doi.org/10.1145/3313831.3376771

[66] Suwen Zhu, Tianyao Luo, Xiaojun Bi, and Shumin Zhai. 2018. Typing on an Invisible Keyboard. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. ACM, New York, NY, USA, Article 439, 13 pages. https://doi.org/10.1145/3173574.3174013

[67] Suwen Zhu, Jingjie Zheng, Shumin Zhai, and Xiaojun Bi. 2019. I'sFree: Eyes-Free Gesture Typing via a Touch-Enabled Remote Control. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. ACM, New York, NY, USA, Article 448, 12 pages. https://doi.org/10.1145/3290605.3300678

A COMMANDS IN EXPERIMENT 2

Thirty commands were used in Experiment 2. Each command could be triggered by one of the two words, as shown in Table 2.

Table 2: The 30 commands and the corresponding words used to trigger them in Experiment 2.

camera, lens	copy, replicate	cut, trim	delete, remove	download, browse
edit, revise	file, folder	keyboard, typing	mail, email	print, publish
rotate, spin	search, find	network, internet	clock, timer	calculator, computer
help, assist	recent, latest	share, exchange	weather, forecast	zoom, enlarge
contact, association	date, calendar	night, dark	bluetooth, connection	photo, gallery
settings, ambience	message, letter	flashlight, torch	sound, voice	notebook, binder