Using Bayes' Theorem for Command Input: Principle, Models, and Applications

Suwen Zhu[1], **Yoonsang Kim**[1], **Jingjie Zheng**[2], **Jennifer Yi Luo**[1,*], **Ryan Qin**[1,*], **Liuping Wang**[3], **Xiangmin Fan**[3], **Feng Tian**[3], **Xiaojun Bi**[1]
[1]Department of Computer Science, Stony Brook University, Stony Brook, NY, United States
[2]Google, Kitchener, Ontario, Canada
[3]Institute of Software, Chinese Academy of Sciences, Beijing, China
{suwzhu, yoonsakim, xiaojun}@cs.stonybrook.edu, jingjiezheng@google.com, {jluo9191, ryanqin15}@gmail.com, wangliuping17@mails.ucas.ac.cn, {xiangmin, tianfeng}@iscas.ac.cn

ABSTRACT
Entering commands on touchscreens can be noisy, but existing interfaces commonly adopt deterministic principles for deciding targets and often result in errors. Building on prior research of using Bayes' theorem to handle uncertainty in input, this paper formalized Bayes' theorem as a generic guiding principle for deciding targets in command input (referred to as "*BayesianCommand*"), developed three models for estimating prior and likelihood probabilities, and carried out experiments to demonstrate the effectiveness of this formalization. More specifically, we applied BayesianCommand to improve the input accuracy of (1) point-and-click and (2) word-gesture command input. Our evaluation showed that applying BayesianCommand reduced errors compared to using deterministic principles (by over 26.9% for point-and-click and by 39.9% for word-gesture command input) or applying the principle partially (by over 28.0% and 24.5%).

Author Keywords
Bayes' theorem; command input; point-and-click; word-gesture shortcuts; touchscreen.

CCS Concepts
•**Human-centered computing** → **Human computer interaction (HCI);** *Interaction techniques;* User studies;

INTRODUCTION
Command input is essential to human-computer interaction. There is no exception in the era of mobile and wearable computing where people regularly issue commands on touchscreens with finger touch or gesture [1, 17, 38, 42]. These input modalities are natural to use, but they inevitably introduce uncertainty. For example, touch input is notoriously known to be noisy and imprecise due to occlusion and the uncertainty of converting a "fat" contact region into a single touchpoint [26, 32, 33, 63, 64, 65]; a recognizer in a gestural command input system may yield erroneous output if the input gestures deviate from the predefined templates [43, 71].

Despite the wide existence of uncertainty, the existing command input methods are ill-positioned for handling it because they often adopt a deterministic principle to decide which command will be issued. For example, to trigger a command with finger touch, the user needs to land the touchpoint precisely within the target boundaries; to input a command with a gesture, the decoded command name should match the exact command label.

Inspired by prior work of adopting Bayes' theorem to decide touch selection target [10], this work incorporates prior probability into the target deciding process, and generalizes the Bayes' theorem as a principle (referred to as *BayesianCommand*) for both point-and-click and gestural command input. Our experiments (explained later) showed incorporating the prior increased command input accuracy by more than 20% (relatively) over using likelihood only [10]. Our work focuses on command input, which also contributes to a large body of research of using a probabilistic framework to handle uncertainty in interaction (e.g., text entry [27], navigation [45], file

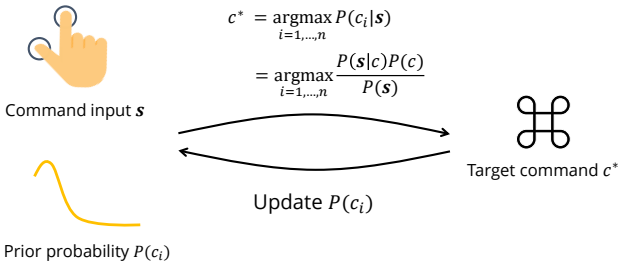


Figure 1. Overview of BayesianCommand. Given an input signal s and a set of n commands $C = \{c_1, \ldots, c_n\}$, the goal of a command input task is to find $c*$ that maximizes $P(c|\mathbf{s})$. BayesianCommand views the input signal s as a random variable carrying likelihood information, and uses Bayes' theorem to combine it with the prior probability $P(c_i)$ to infer $P(c|\mathbf{s})$. The target command information is then used to update the prior probability model.

*Jennifer Yi Luo and Ryan Qin were high school interns supervised by Suwen Zhu and Xiaojun Bi.

CHI '20, April 25–30, 2020, Honolulu, HI, USA

ISBN 978-1-4503-6708-0/20/04…15.00
DOI: https://doi.org/10.1145/3313831.3376771

retrieval [46], estimating capacitive sensing uncertainty [59], inferring interaction intention for probabilistic widgets [14]).

BayesianCommand sets out from a probabilistic perspective to interpret the ambiguity in command input: the input signals are viewed as a random variable that carries likelihood information of the target command. The posterior belief is formed accordingly via Bayes' theorem. The candidate with the highest posterior probability should be decided as the target. This information is in turn used to update the prior probability model for future command input. Figure 1 provides an overview of BayesianCommand. Because BayesianCommand is algorithmic and requires no visual change of the layout, it is advantageous to frequency-based layout rearrangement and less likely to slow users down [20].

Overall, we made the following three contributions: (1) We formally described how to use Bayes' theorem for command input, and established that it should be the principle for deciding the target command; (2) We proposed three models for applying BayesianCommand: a prior probability model, a dual-Gaussian likelihood model for point-and-click input, and a two-step likelihood model for recall-based gestural command input; (3) We conducted experiments to demonstrate the effectiveness of BayesianCommand over using the existing deterministic principles or applying them partially.

RELATED WORK

This work is related to handling uncertainty in user input, touch pointing and gestural command input technologies.

Handling Uncertainty in User Input

Probabilistic frameworks have been proposed to deal with the uncertainty in input processes, such as considering the input as a continuous control process in which the system continuously infers a distribution over potential user goals [13, 67, 70], or carrying the uncertainty of input forward all the way through the interaction [61, 62]. Other examples include Dasher [66], which used probabilistic models to adapt screen layouts, and Semantic pointing [12], which adapted the control-to-display ratio according to cursor distance to nearby targets. Bayes' theorem has also been adopted to reduce uncertainty in interaction, such as the statistical decoding algorithm of soft keyboards [27], and the Bayesian Information Gain (BIG) framework [45, 46].

Distinct from the previous work, we focus on command input, adding to the vast body of research of using probabilistic frameworks to handle input uncertainty. BayesianCommand uses Bayes' theorem to incorporate the previous command input history to improve input accuracy. Although previous research has explored adapting the menu visuals (e.g., Morphing menu [16]) to accommodate the command frequency, BayesianCommand is algorithmic and requires no visual change, making the UI visual consistent to users throughout the interaction.

Understanding and Improving Finger Touch

There has been extensive research on understanding and improving touch pointing accuracy. On a capacitive touchscreen, a touchpoint is converted from the finger's contact region with noise and uncertainty in the converting process. Factors such as hand posture [14, 26], finger angle [32, 33], and body movement [25, 55] may affect the size and shape of the contact region, unintentionally altering the touch position. The lack of feedback on where the finger lands due to occlusion (the "fat finger" problem) further exacerbates the issue [32, 33, 63, 64, 65]. Previous research has explored various approaches to improve touch accuracy. Examples include compensating for the offset caused by different finger input angles [32, 33] or location on screens [31], displaying the touch location in a non-occluded area [64], and using the back of the device for selection [68, 69]. Others also explored using various finger gestures to assist target selection, including crossing [2, 15, 51, 52, 58], sliding [14, 54, 72, 73], rubbing [57, 60], circling [34], and multi-touch gestures [8].

Our first application of using BayesianCommand to improve point-and-click command input is in particular related to Bayesian Touch Criterion (BTC) [10]. The main difference is that BayesianCommand involves both prior and likelihood probability calculation, while BTC ignored the prior probability and only used the likelihood. We compared BayesianCommand with BTC in our user study. We showed that incorporating prior is an essential step toward truly adopting Bayes' theorem, and it substantially improves the touch accuracy over BTC. Apart from point-and-click input, our second application of gestural command input is different from BTC [10].

Word-Gesture Command Input

Gestural input has been widely explored as a command input method on touchscreen devices, thanks to the human's ability in memorizing pictorial information [56]. It has been adopted in marking menu [39, 40, 41] and its variants [5, 6, 22, 23, 75, 76], gesture-based interfaces [7, 24, 42, 47, 48, 49, 50], and multi-touch gesture frameworks [35, 36].

To assist users in memorizing the mappings between commands and gestures, previous researchers have explored using word-gestures [37, 74] for command input – entering a command by gliding finger over letters in the command name on a virtual keyboard. Word-gesture was initially invented for text entry on touchscreen devices [37, 74], which was later extended as a method for command input. For example, Command Strokes [38] and CommandBoard [1] support triggering a command by drawing its word-gesture on a soft keyboard, and HotStrokes [17] supports word-gesture command input on a laptop trackpad.

The existing gestural command input systems (e.g., [1, 17, 38] often adopt a deterministic principle to decide the target command: the decoder matches the input gesture with the predefined gesture template of each command candidate; the candidate with highest matching score is the target command. It has little room for handling uncertainty and would result in errors if the input gesture deviates greatly from the template or some commands share the similar predefined templates. In this paper we proposed BayesianCommand to replace the typical deterministic principle to handle such uncertainty.

BayesianCommand: A BAYESIAN PERSPECTIVE ON COMMAND INPUT

From a Bayesian perspective, a command input task can be described as follows: assuming $C = \{c_1, c_2, \ldots, c_n\}$ is a set of n available commands, given the input signal $\mathbf{s}$, the goal of a command input task is to find c^* in C that maximizes $P(c|\mathbf{s})$. According to Bayes' theorem, it can be calculated as:

$$c^* = \arg\max_{c \in C} P(c|\mathbf{s}) = \arg\max_{c \in C} \frac{[P(\mathbf{s}|c)P(c)]}{P(\mathbf{s})}. \quad (1)$$

Assuming $P(\mathbf{s})$ is a constant across c (because $\mathbf{s}$ is a fixed value for a given input), we can further simplify Equation (1) to:

$$c^* = \arg\max_{c \in C} [P(\mathbf{s}|c)P(c)], \quad (2)$$

where $P(c)$ is the prior probability of c being the intended command without the observation of $\mathbf{s}$, and $P(\mathbf{s}|c)$ describes how probable $\mathbf{s}$ is if the intended target is c (the likelihood). We refer to this principle as BayesianCommand in this paper.

Obtaining the prior and likelihood is the key to apply BayesianCommand. We developed one prior probability model, and two likelihood models: a dual-Gaussian likelihood model for point-and-click and a two-step likelihood model for command shortcut. The dual-Gaussian likelihood model was inspired by the dual Gaussian distribution hypothesis [9, 10]. Note that these models represent only one approach of estimation. We use them to establish that BayesianCommand is the principled framework for command input.

PRIOR PROBABILITY MODEL

We first developed a model to predict $P(c)$ – the prior probability of the candidate c being the intended target – from the command input history.

In the prior probability model, we assume that the distribution of the intended command among candidates is not entirely random, and the command input history is observable. We formed this assumption based on the findings that the patterns of menu selection [16, 44], command triggering [3, 19], and smartphone app launching [53] are not random and often follow certain distributions (e.g., *Zipfian* distributions). These are all scenarios involving frequent command input.

Before deriving the model, we define two criteria that the model should satisfy:

(1) Without observing any selection history, each candidate is equally probable as the target.

(2) With a large number of observations, $P(c)$ approximates the frequency that the candidate c was selected as the target in the past.

We propose the frequency model as follows. Assuming we observe that the candidate c_i has been selected t_i times in the past as the target, $P(c_i)$ is calculated as:

$$\forall i, P(c_i) = \frac{k + t_i}{k \cdot n + \sum_{i=1}^{n} t_i}, \quad (3)$$

where n is the number of available commands (e.g., the number of items in a menu), and k is the update rate, a positive constant which determines how fast $P(c_i)$ will be learned from the selection history.

The proposed model (Equation (3)) satisfies aforementioned criteria (1) and (2). If no selection history is observed, i.e., $t_i = 0, i \in [1, n]$, Equation (3) shows $P(c_i) = \frac{1}{n}$. It indicates that each candidate is equally probable as the target. On the other hand, if we have a large number of observations on selection history (i.e., $t_i \gg k \cdot n$ and $t_i \gg k$, $i \in [1, n]$)), Equation (3) shows $P(c_i) \approx \frac{t_i}{\sum_{i=1}^{n} t_i}$, which is the frequency of c_i being the target in the past.

The update rate k in the model controls the balance between two extreme views on calculating $P(c_i)$:

(A) $P(c_i)$ is identical to the frequency of c_i being the target in the past.

(B) all the candidates are equally probable as the target.

If $k = 0$, $P(c_i) = \frac{t_i}{\sum_{i=1}^{n} t_i}$, which is the view (A). If $k \to +\infty$, $P(c_i) \approx \frac{1}{n}$, which is the view (B). A positive k controls the weights between these two views. Later we explain how we used a simulation-based approach to determine an optimal k in our applications.

LIKELIHOOD MODELS

We have developed two likelihood models: a dual-Gaussian likelihood model for point-and-click command input, and a two-step likelihood model for recall-based gestural command input. Because each of the likelihood models is tightly connected to the specific command input method, we describe how to obtain them when describing applications.

After obtaining prior probability and likelihood, we can apply BayesianCommand to decide the target command. Algorithm 1 shows how the BayesianCommand principle works.

Algorithm 1 BayesianCommand

Input: s – the input signal s,
  C – a set of command candidates $\{c_1, \cdots, c_n\}$
Output: the target command $c*$
for $i = 1, 2, \cdots, n$ **do**
  obtain prior probability $P(c_i)$ from Equation (3)
  calculate $P(s|c_i)$ from the likelihood model
select $c* = \arg\max P(s|c_i)P(c_i)$ as the target command
update prior probability $P(c_i)$ for each c_i based on Equation (3), given that $c*$ is the selected command.

APPLICATION 1: APPLYING BayesianCommand TO POINT-AND-CLICK COMMAND SELECTION

We first applied BayesianCommand to improve the command input accuracy on a touch-based point-and-click interface: triggering a command by touch pointing the corresponding icon, button, or menu item. We expect that BayesianCommand will improve the command input accuracy over the typical *boundary criterion*, which decides the target by examining whether the touchpoint falls within the target boundary.

To apply BayesianCommand to point-and-click command selection, we used the prior probability model (Equation (3)) to estimate $P(c)$. Next, we explain how to obtain $P(\mathbf{s}|c)$ for point-and-click interfaces.

Dual-Gaussian Likelihood Model for Point-and-Click

We adopted a dual-Gaussian likelihood model to calculate $P(\mathbf{s}|c)$, assuming that the touchpoints approximately follow a Gaussian distribution [4, 31, 32]. Assuming for a 2-dimensional target we observe a touchpoint $\mathbf{s}$ as (s_x, s_y), $P(\mathbf{s}|c)$ can be calculated as:

$$P(s|c) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left[-\frac{z}{2(1-\rho_i^2)}\right], \quad (4)$$

where

$$z \equiv \frac{(s_x-\mu_x)^2}{\sigma_x^2} - \frac{2\rho(s_x-\mu_x)(s_y-\mu_y)}{\sigma_x\sigma_y} + \frac{(s_y-\mu_y)^2}{\sigma_y^2}. \quad (5)$$

(μ_x, μ_y) is the target center, σ_x and σ_y are the standard deviations of users' touchpoints, and ρ is the correlation coefficient between x and y. We followed the next two steps to estimate the parameters of Equations (4) and (5).

First, we assumed that the center of touchpoint distribution (μ_x, μ_y) co-locates with the center of the target. Previous research showed that (μ_x, μ_y) has only a small offset from the target center, and the magnitude and direction of the offset are affected by various factors including the target position on the screen, users' postures, and finger angle, etc. [4, 31, 32, 67, 77]. Without further knowledge on these factors, we assume (μ_x, μ_y) is located at the target center. Similarly, previous research also showed the correlation coefficient (ρ) between x and y largely depends on a variety of factors such as on-screen location, hand posture, and finger angle. Similar to the approach adopted in Bi and Zhai [11], we assume $\rho \approx 0$ without further knowledge of these factors.

Second, we adopted the dual Gaussian distribution hypothesis [9, 10] to estimate σ_x and σ_y. For a point-and-click interface, the dual distribution hypothesis [9, 10] states that the variance of touchpoints (σ) has a linear relationship to d^2:

$$\sigma^2 = \sigma_r^2 + \sigma_\alpha^2 = \alpha \times d^2 + \sigma_\alpha^2, \quad (6)$$

where α and σ_a are empirically determined parameters, and d is the target size.

Parameterizing the dual-Gaussian Likelihood Model

Following the procedure reported in the previous research [10], we conducted a target acquisition study to obtain α and σ_a values for Equation (6).

We recruited 36 participants (12 female) aged between 19 and 37 (average 25.4±4.2). Each participant was instructed to naturally select a circular target, which randomly appeared on a Nexus 5X touchscreen device. The study included four levels of target size (diameter): 8, 12, 16, 20 mm, each with 20 trials. To avoid over-fitting, we randomly divided the data into two sets: 29 (~ 80%) participants as the training set and the rest as the test set. Both data sets included a mix of two postures (index finger, thumb).

We established the touch model of the training set following the procedure described in [10]. More specifically, we first calculated the mean and standard deviation of the touchpoints relative to the target center. As conventional in Android and iOS, we assume the positive x-direction is right, and the positive y-direction is down. Table 1 shows the touch model parameters (in mm) of the training set data.

d	μ_x	μ_y	σ_x	σ_y
8	0.472	0.327	1.372	1.598
12	0.648	0.348	1.756	2.010
16	0.628	0.411	1.843	2.350
20	0.973	0.348	2.138	2.451

Table 1. Touchpoint distribution for different target sizes. All units are in *mm*. The target center is (0,0). d is the diameter of the target. μ_x, μ_y are the mean of the touchpoints. σ_x and σ_x are the standard deviations of the touchpoints.

We then ran linear regression for the variance of x and y directions against d^2. The estimations are shown in Figure 2. The α and σ_a values serve as the parameters for Equation (6). To verify the trained parameters, we tested them on the σ values on the test dataset, the mean (SD) RMSE were 0.10 (0.11) *mm* on σ_x and 0.12 (0.04) *mm* on σ_y across different d. This confirmed the validity of the model.

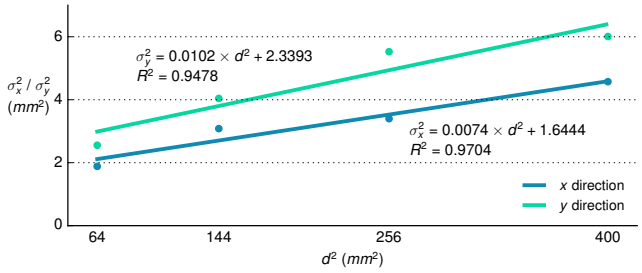


Figure 2. Regression between the variance in x directions (σ_x^2) / y direction (σ_y^2) and the target width (d^2).

In this section, we provide the dual-Gaussian likelihood model for point-and-click command input by building a touch model that predicts the probability of observing a touchpoint $\mathbf{s}$ given c as the intended command. This touch model not only serves as the likelihood function for BayesianCommand but can also be used to generate touchpoints in the following simulation study in which we determined how fast the prior probability model should be updated from the command input history.

Determining the Update Rate of Prior Probability Model

After obtaining the prior probability and likelihood models, we investigated how fast the prior probability $P(t_i)$ will be updated from the selection history. In other words, we decided the optimal k value in Equation (3) via a simulation study.

The simulation worked as follows. We first designed a 6 by 4 touchscreen grid layout for command selection (Figure 3). Each cell in the grid corresponded to a command candidate. We then implemented BayesianCommand as the principle for deciding the target on this grid interface, using the previously

described prior probability model and likelihood model. We implemented a set of BayesianCommand-based criteria with different k values in the prior probability models. We used the touch models of test set users to generate the touchpoints, fed the touchpoints into this grid layout and evaluated the accuracy of different BayesianCommand models to determine which k value led to optimal performance.

We ran the simulation using the data collected in the previous study. We used the training set to train a touch model which served as the likelihood model in BayesianCommand (Figure 2). We developed independent touch models for each user in the test set to generate the touchpoints for testing.

On this grid layout, we assumed the target frequency follows the Zipfian distribution [79]:

$$f(l; s, N) = \frac{1/l^s}{\sum_{n=1}^{N}(1/n^s)}, \quad (7)$$

where N is the number of elements, $l \in \{1, 2, \ldots, N\}$ is the rank of the element, and s is the value of the exponent characterizing the distribution. We randomly picked 12 square targets from a grid layout (Figure 3) and simulated two different distributions with exponent $s = 1$ and $s = 2$, based on 600 total selections. The generated frequencies were (216, 106, 98, 79, 52, 25, 7, 6, 4, 4, 2, 1) for $s = 1$ and (430, 142, 14, 3, 2, 2, 2, 1, 1, 1, 1, 1) for $s = 2$. We assumed these 12 frequencies showed how frequently a target would be the intended command, and assigned these 12 frequencies to the selected 12 targets.

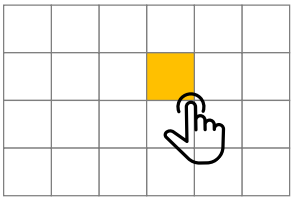


Figure 3. The grid layout used in the simulation. The yellow block shows the simulated target. The finger illustrates the simulated touchpoint.

Seven target sizes (4, 5, 6, 7, 8, 9, and 10mm) were tested. We ran the simulation for every user in the test set separately. The target order was randomized. In each simulation trial, we picked one candidate as the target, and generated a touchpoint for selecting this target following the test user's individual touch model. Given the touchpoint location, we then determined the selected target using BayesianCommand with different k values in the prior probability models. We repeated the procedure five times.

In total, the simulation included: 2 Zipfian distributions × 7 target sizes × 7 test users × 600 trials × 5 repetitions = 294000 simulation trials.

To determine which k should be used in the prior probability model, we compared the following k values when applying BayesianCommand:

(1) optimal k. We searched for the optimal k in the prior probability model by initializing k to 0.1, and increasing it to 20 with a step length of 0.1. The k that led to the highest accuracy was optimal.

(2) $k = 1$. We used $k = 1$ across users and conditions. We discovered that $k = 1$ performed well in pilot simulation runs and would like to see if it could be generalized.

Results

We calculated the target acquisition accuracy of each repetition as the total number of correct selections divided by the total number of selections averaged across the users in the test set. Figure 4 shows the average accuracy over the five repetitions.

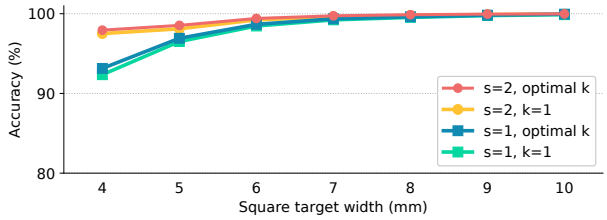


Figure 4. Average target acquisition accuracy by target size for different target frequency (Zipfian distribution with $s = 1$ or $s = 2$) and k.

The optimal value of k varied across different runs, but generally fell within the range of $[0.5, 3]$. As shown in Figure 4, when $k = 1$, the average accuracy was close to the optimal accuracy. Therefore, we chose $k = 1$ in the prior probability model (Equation (3)) and used this value for implementation. Note that the choice of k is specific to our particular application. Different values should be selected depending on the actual scenario.

After determining the update rate (i.e., the k parameter), we conducted a study to evaluate BayesianCommand for point-and-click command input, using the proposed prior probability model and the dual-Gaussian likelihood model.

Experiment I: Evaluating BayesianCommand for Point-and-Click Command Input

The purpose of the study was to evaluate BayesianCommand for point-and-click command input. We expected BayesianCommand to outperform the typical boundary criterion because BayesianCommand was a more principled approach to handle the ambiguity in touch pointing input. We were also interested in comparing BayesianCommand with the BTC [10]. As explained in the related work section, BTC ignores the prior probability and only uses the likelihood probability to decide the target.

Participants and Apparatus

18 adults (4 females) aged between 21 and 35 (average 27.3 ± 3.3) participated in the study. 16 participants were right-handed. The self-reported average usage time of mobile phones was 24.5 hours per week. We used a Ticwatch S Smartwatch with a 45mm diameter screen in the study (Figure 5a).

Experiment Setup

The study was a within-subject design. There were 2 independent variables: target size and target deciding principle. We evaluated two target sizes: 3mm and 4mm square targets on a 4 by 6 grid layout. The target deciding principles included three levels:

- BayesianCommand. It used BayesianCommand (Algorithm 1) to decide the target. We used Equation (3) as the prior probability model and Equation (6) (Figure 2 for parameters) to obtain the likelihood value. We chose $k = 1$ in the prior probability model according to the previously described simulation study.

- BTC [10]. BTC uses only the likelihood function to decide the target command. As with BayesianCommand, BTC also used the dual Gaussian distribution hypothesis [9, 10] to obtain the likelihood value. To be consistent with the BayesianCommand experimental condition, we used the same parameters used in BayesianCommand (Figure 2) for BTC.

- Boundary Criterion. This is the commonly adopted criterion that decides the target command by examining whether the touchpoint falls within the target boundaries. It served as a baseline in our experiment.

In the BayesianCommand and BTC conditions, we used the same touch model obtained from the previous study in the likelihood model. Except for the form factor, the two devices used capacitive touch screens and were both running Android OS, i.e., the underlying mechanism to convert the finger touch to a touchpoint was the same. We assumed the previously developed touch model was valid on our testing device.

We designed a point-and-click command input task. The item corresponding to the target command was highlighted in yellow. Participants were instructed to select the target item as fast and accurately as possible. When a selection was made, the selected item would be highlighted with a blue background. A trial was completed if the selection was correct or three failed attempts were made.

(a) (b)

Figure 5. The setup of Experiment I. (a) shows a participant selecting a 4mm target, and (b) shows the application with 3mm square targets. The ones highlighted by yellow were the targets tested in the experiment.

We randomly selected 12 items as targets. We used the same set of targets across participants and conditions. The target positions were fixed, as shown in Figure 5b. Target item frequencies were generated according to Zipfian distribution with exponent $s = 1$ based on 30 selections. The generated frequencies, i.e., the number of occurrences, were (7, 5, 4, 4, 2, 2, 1, 1, 1, 1, 1, 1). The frequency assignments were randomized across participants and conditions. Participants were not informed of the frequency distribution of the items or the position of the most frequent items.

We balanced the frequency assignments on the target items across all participants and conditions. Each target item was assigned to each frequency an equal number of times to ensure that the same total number of selections was collected for each target. The order of the targets within each block was randomized. A similar strategy was used in [3, 28].

Before the formal study, participants were introduced to the task and performed a warm-up session of 5 trials. Each condition contained two blocks, each with 30 trials. Every participant performed the task three times in a row, using a different target deciding principle each time. The order of the three principles was fully counterbalanced across the 18 participants.

In total, the study included: 18 participants $\times$ 3 principles $\times$ 2 target sizes $\times$ 60 trials = 6480 trials.

Results

Error rates. This metric measures the ratio of the number of incorrect selections over the total number of trials. The average error rates by target deciding principle are shown in Figure 6.

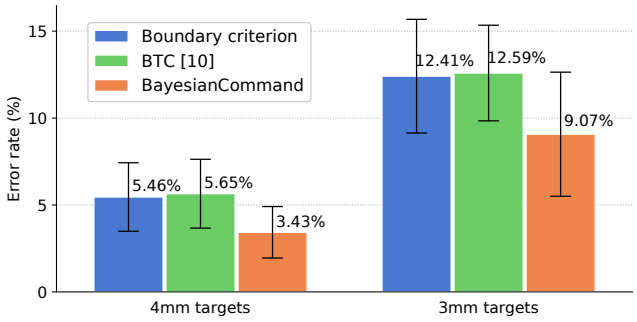


Figure 6. Average error rate (95% confidence interval) of the three target deciding principles on 3mm and 4mm targets.

BayesianCommand reduced the error rates: on 4mm targets, it reduced the error rate by 37.2% and 39.3% compared to the boundary criterion and BTC; on 3mm targets, the error rate reduction was 26.9% and 28.0% respectively. ANOVA showed there was a significant main effect of the target deciding principle on the error rates ($F_{2,34} = 7.98, p < .005$). Pairwise comparisons with Bonferroni adjustment showed that the difference was significant between BayesianCommand vs. BTC ($p = .004$) and between BayesianCommand vs. boundary criterion ($p = .017$). The 4mm targets were less error-prone and easier to select than the 3mm targets. ANOVA showed the differences were significant for target size ($F_{1,17} = 39.93, p < .005$). We did not observe a significant interaction effect of target deciding principle $\times$ target size ($F_{2,34} = 0.33, p = .72$).

Target acquisition time. We compared the average target acquisition time, which was the elapsed time from a target being highlighted on the screen to the time the participant made the first selection. We only considered the first attempt in every trial, regardless of whether it was correct or not.

Target size	Boundary criterion	BTC [10]	BayesianCommand
4mm	0.74 ± 0.25	0.78 ± 0.26	0.71 ± 0.15
3mm	0.86 ± 0.26	0.88 ± 0.24	0.94 ± 0.44

Table 2. Average target acquisition time in seconds.

The target size had a main effect on the target acquisition time ($F_{1,17} = 6.98, p = .017$). We did not observe a main effect of the target deciding principle ($F_{2,34} = 0.30, p = .75$) or any interaction effect ($F_{2,34} = 1.45, p = .25$). As shown in Table 2, using different target deciding principles had little effect on the target acquisition time.

Subjective feedback. We used a subset of NASA-TLX [30] questions to measure the perceived workload of the task, including mental demand, physical demand, and effort. The rating was from 0 to 10. The lower the rating the better.

For 4mm targets, the median ratings were 4 (mental demand), 3 (physical demand), and 3 (effort) for BayesianCommand; 3, 4, 4 for the boundary criterion, and 3.5, 4, 4 for BTC. For 3mm targets, the median ratings were 4, 5, 5 for BayesianCommand; 5, 5, 5 for the boundary criterion, and 5, 5, 6 for BTC. BayesianCommand was perceived slightly less mentally demanding than the other two principles on 3mm targets.

Discussion

The empirical study showed that BayesianCommand outperformed the boundary criterion and BTC. Using BayesianCommand substantially reduced the touch pointing error rate for both large and small menu targets. The reduction was especially remarkable for small targets: around 26% over both boundary criterion and BTC. It also showed that learning the prior probability distribution and combining it with the likelihood function outperforms using the likelihood function alone. Since BayesianCommand is algorithmic, these improvements were achieved without altering any UI layout, which was advantageous to frequency-based menu adaptation (e.g., morphing menu [16]), and thus less likely to slow users down or reduce user satisfaction [20].

BTC had almost identical error rates to the boundary criterion. According to the definition of BTC (Equation (1) in [10]), when the target sizes are equal, BTC is equivalent to comparing the distance from the touchpoint to the target center (touchpoint-to-center distance). Since the targets were of the same size and were arranged in a grid with no gaps between them in our experiment, BTC was equivalent to boundary criterion: the item whose boundary contains the touchpoint is also the target that has the shortest distance to the touchpoint. Although we only allowed 3 failed attempts per trial, our investigation showed it had minor effects on the overall error rates. 16 participants could correctly finish all trials within 3 attempts in all conditions. The rest two failed 0.56% (= 4/720) of the trials 3 times, which were for 3mm targets in the BayesianCommand condition. These 2 participants were able to select the targets correctly when the same trials repeated, indicating that these items remained accessible for them.

How can designers or developers leverage the benefits of BayesianCommand? Many applications and software have collected usage patterns of menus, buttons, and commands, e.g., the command usage frequency of Microsoft Word 2003 [29]. These accumulated frequencies and patterns could serve as the prior probability for adopting BayesianCommand; the system can then adapt the prior probability as a user is interacting with the system. If no prior command history is available, the system can assume every command is equally probable and learn the distribution probability as more actions are observed. We would also like to point out that BayesianCommand works under the assumption that the command input distribution model can be established. It might not show significant benefits for some applications if their command frequency model is not that obvious (e.g., Maps).

APPLICATION 2: APPLYING BayesianCommand TO WORD-GESTURE SHORTCUTS

In this application, we investigated how to apply BayesianCommand for word-gesture shortcuts – entering a command by drawing the word-gesture [37, 74] of the command name (e.g., Figure 7). We first propose a two-step likelihood model for gestural input and combine it with the previously proposed prior probability model. A user study showed using BayesianCommand outperformed the existing deterministic principle, which selects command simply based on the highest matching score from a gesture decoder.

Note that the two-step likelihood model is not a gesture decoder; it is a model that uses a gesture decoder and combines the decoding outcomes with available command candidates to estimate $P(\mathbf{s}|c)$, where $\mathbf{s}$ is an input gesture and c is a command. It is independent of a gesture decoder. In this application, we used the i'sFree gesture decoder [78] as an example, but it can be replaced with other decoders such as $SHARK^2$ [37].

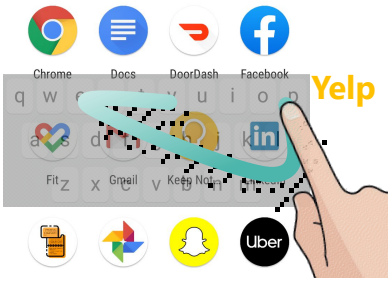


Figure 7. Launch *Yelp* with word-gesture shortcuts: drawing the word-gesture (in green) of the word *Yelp* on an imaginary Qwerty keyboard. We use the i'sFree gesture decoder [78] in this example, so the keyboard is invisible to the users (illustrated as semi-transparent).

Two-Step Likelihood Model for Word-Gesture Shortcuts

To develop the model, we first view the decoding process, i.e., the procedure of mapping an input gesture $\mathbf{s}$ to a command c as a two-step process:

- Step-1: $\mathbf{s}$ is first decoded into a word w by a gesture decoder (e.g., $SHARK^2$ [37, 74] or i'sFree decoder [78]).
- Step-2: w is mapped to a specific command c. Note that a user may trigger a command with different words. For example, to launch a clock application, users could input *clock*, *time*, *timer*, or *watch*.

If we view **s**, w, and c as random variables, the following graphical model describes their dependencies (Figure 8):

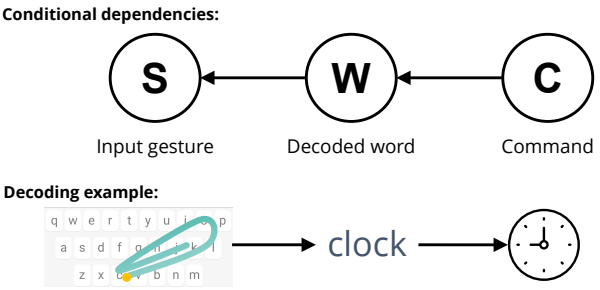


Figure 8. A graphical model showing the conditional dependencies between the input gesture s, decoded word w, and command c. The example shows the process of triggering the command "clock" with a word-gesture. The word-gesture is entered on a keyboard, and the yellow dot illustrates the start of the gesture.

We developed the two-step likelihood model based on this graphical model. According to the law of total probability, we can get $P(\mathbf{s}|c)$ as:

$$P(\mathbf{s}|c) = \sum_{i=1}^{N} P(\mathbf{s}, w_i|c) = \sum_{i=1}^{N} P(\mathbf{s}|w_i, c)P(w_i|c), \quad (8)$$

where **s** is the input gesture, w_i is a decoded word candidate from a gesture typing decoder, N is the total number of decoded word candidates, and c is a command candidate.

The graphical model (Figure 8) suggests that c and **s** are conditional independent given w. Therefore, Equation (8) can be further expressed as:

$$P(\mathbf{s}|c) = \sum_{i=1}^{N} P(\mathbf{s}|w_i, c)P(w_i|c) = \sum_{i=1}^{N} P(\mathbf{s}|w_i)P(w_i|c). \quad (9)$$

Equation (9) is our two-step likelihood model. As shown, the key of using this model is to obtain $P(\mathbf{s}|w_i)$ and $P(w_i|c)$. These two terms can be calculated as follows.

The term $P(\mathbf{s}|w_i)$ represents the probability of observing the input gesture **s** if w_i is the target word. From a gesture typing decoder's perspective, it is the spatial score of w_i given **s** is the input gesture [37, 74]. In this research, we adopted the eyes-free gesture decoder [78] to obtain it. We swapped the original language model used in the eyes-free gesture decoder [78] with the command set $C = \{c_1, c_2, \ldots, c_n\}$, because our goal was to predict an available command in a command set, rather than as a general text entry method.

The term $P(w_i|c)$ represents the probability of inputting the word w_i if the c is the intended command. Since a command might be triggered by different words (e.g., launching a clock with *clock*, *timer*, or *watch*), we calculate $P(w_i|c)$ as follows. For a given command c, we first form a set of words corresponding to it: $M = \{m_0, m_1, m_2, ..., m_K\}$ from a thesaurus (e.g., thesaurus.com), where m_i is a valid word for triggering c. If a decoded word candidate w_i does not belong to this set, we assume $P(w_i|c) = 0$. Otherwise, $P(w_i|c) = \frac{1}{K}$, assuming that each word in this thesaurus has equal probability for triggering command c.

Equation (9) is the two-step likelihood model for decoding word-gesture command input. After obtaining $P(w_i|c)$ and $P(\mathbf{s}|w_i)$, we can then use it to calculate $P(\mathbf{s}|c)$. Together with the prior probably model (Equation (3)), we can apply BayesianCommand (Algorithm 1) to decide the target command for word-gesture shortcuts. Note that this is only one design option for the likelihood model. Our purpose is not to prove it is superior over other options. Instead, we used it as an example to demonstrate how to use BayesianCommand as the principled way to decide target command in gestural command input. There could be other alternatives. For example, we may use the decoding likelihood $P(\mathbf{s}|w_i)$ from a gesture decoder to directly approximate $P(\mathbf{s}|c)$, assuming there is a one-to-one mapping between w and c. We adopted the two-step model because it reflects the gesture command decoding procedure, and offers more flexibility. For example, it can model situations where different words w_i can trigger the same command c, and the same word w can trigger multiple commands (e.g., depending on the application context) by including the same w in multiple M sets.

Experiment II: Evaluating BayesianCommand for word-gesture shortcuts

We conducted a user study to evaluate using BayesianCommand for word-gesture shortcuts. We compared BayesianCommand with the typical deterministic strategy for deciding target command.

Participants and Apparatus

18 adults (4 females) aged between 23 to 31 (average 26.9±2.5) participated in the study. The self-reported average usage time of mobile phones was 30.1 hours per week. 17 participants were right-handed. The median of self-reported familiarity with Qwerty layout (1: not familiar at all, 5: very familiar) was 4.5. The median familiarity with gesture typing was 3. A Google Pixel running Android 9.0 was used for the study, as shown in Figure 9b.

Experiment Setup

The study was a within-subject design. The independent variable was the command deciding principles with three levels:

- BayesianCommand: we applied BayesianCommand (Algorithm 1) as the principle to decide the target command, using Equation (3) to calculate prior probability and the two-step likelihood model in Equation (9) to calculate likelihood. Similar to Experiment I, we chose $k = 1$ in the prior probability model.

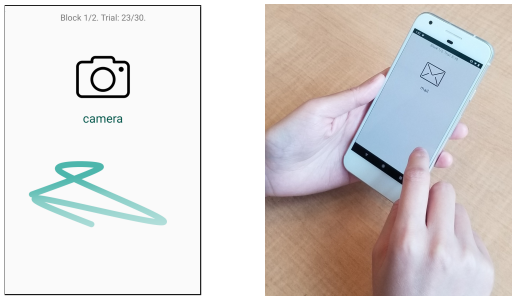
- Likelihood-only: the command candidate with the highest likelihood value (the two-step likelihood model in Equation (9)) is the intended target. It uses likelihood value only. We included this condition to understand how much performance gain in BayesianCommand was provided by the prior probability, and how much gain was provided by the likelihood function. This approach can also be viewed as using BayesianCommand but assuming all the command candidate has equal prior probability: under this assumption $P(c|\mathbf{s})$ will be determined by the likelihood $P(\mathbf{s}|c)$ only.

- Deterministic approach. This is the typical target deciding principle for gestural command input. The gesture decoder used a set of available command names as the dictionary and matched the input gesture with the words in this dictionary. The word with the highest matching score was the intended command.

We used the same gesture decoder [78] across all three conditions. We swapped the language model in the original decoder with the command set used the study (including all the trigger words for each command in Appendix A). The composition of the command set is explained in detail later.

Before the study, participants were shown the 20 commands and their corresponding graphical representations. Participants needed to memorize ≥ 80% of the commands before they could proceed to the formal study: they had to recall at least one of the trigger words of the commands. This procedure ensured that the results wouldn't be affected by participants' familiarity with the commands, or any external cause other than the three principles.

For each trial, an icon was first displayed on the screen as the target command. The participants then gestured the word in the white space below it to trigger the command. The input command name was shown to the participants after the finger lifted off from the screen, regardless of whether it's the intended command or not, as shown in Figure 9a. A trial was completed if the input command was correct or three failed attempts were made. For each condition, participants first performed a warm-up session of two trials, followed by 60 trials divided into two blocks. Participants were allowed to take a short break after the completion of each block. Each participant performed the task three times, with different target deciding principles each time. The orders of three target deciding principles were fully counterbalanced across participants.



(a) (b)

Figure 9. (a): the application for Experiment II. The user draws a word-gesture command, then the target command will be shown on the screen. (b): experiment setup.

calculator	delete	keyboard	rotate
camera	download	mail	search
clock	edit	network	share
copy	file	print	weather
cut	help	recent	zoom

Table 3. List of the 20 commands. The underlined commands were tested in the experiment.

A subset of 12 commands was picked as the targets. The same set of commands were used across participants. We used the same item frequencies as in Experiment I, i.e., the number of occurrences for the commands was (7, 5, 4, 4, 2, 2, 1, 1, 1, 1, 1, 1). Participants were not informed of the frequency distribution of the items. The rest of the experiment design is similar to Experiment I. For each command, a set of 10 additional words for triggering this command was created from [18]. The list of commands is shown in Table 3. The command set included 20 commands. Each command has 11 corresponding trigger words (10 synonyms and the command name). The command set includes 220 words in total, which was incorporated into the decoder used in the study.

In total, the study included: 18 participants × 3 principles × 60 trials = 3240 trials.

Results

Error rates. This metric measures the ratio of the number of incorrect gesture inputs over the total number of trials. The average error rates are shown in Figure 10.

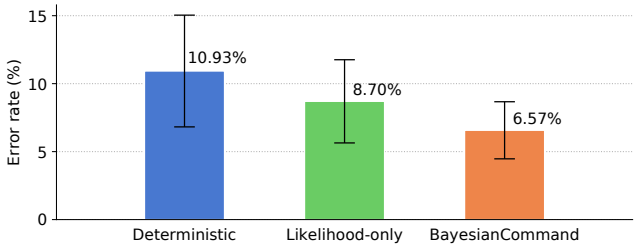


Figure 10. Average error rate (95% confidence interval) of the three principles for word-gesture shortcuts.

BayesianCommand lowered the error rate by 39.9% and 24.5% compared to the deterministic and the likelihood-only approaches. ANOVA showed a significant main effect of the command deciding principle on the error rates ($F_{2,34} = 5.56, p < .01$). Pairwise comparisons with Bonferroni adjustment showed that the difference was significant between BayesianCommand vs. deterministic strategy ($p = .01$), but not for BayesianCommand vs. likelihood-only ($p = .13$) or likelihood-only vs. deterministic strategy ($p = .54$).

Command triggering time. We compared the average command triggering time, which was the elapsed time from a target command icon being shown on the screen to the completion of a gesture command. The average command triggering time was 2.91 ± 1.32 seconds for the deterministic strategy, 2.83 ± 0.98 seconds for the likelihood-only approach, and 2.98 ± 1.35 seconds for BayesianCommand. We did not observe a main effect of the principle ($F_{2,34} = 0.19, p = .83$). This result also indicates that using different principles had little effect on the overall command triggering time.

Use of trigger words. We examined the trigger words of each command in the deterministic condition. We counted the number of unique trigger words (i.e., the decoded word from the gesture recognizer) of each command when it was successfully triggered. We excluded the BayesianCommand

and the likelihood-only conditions as they used probabilistic approaches, and the decoded word did not always correspond to the command label. The average number of trigger words was 2.92 (SD=1.38) across the 12 tested commands. This result supported the validity of the graphical model (Figure 8) and our hypothesis that multiple trigger words could be used for a command.

Subjective feedback. A subset of NASA-TLX [30] questions was used to measure the perceived workload of the task. The range of the ratings was 1 to 10 (the smaller the rating, the better). The median ratings were 4 (mental demand), 2.5 (physical demand), and 3 (effort) for BayesianCommand; 5, 4, 5 for the deterministic method, and 4.5, 4, 5 for likelihood-only. BayesianCommand was perceived less demanding than the other two principles in all questions.

Discussion

The results showed that BayesianCommand effectively improved the input accuracy for word-gesture shortcuts. It reduced the command triggering error rate by 39.9% compared to the deterministic method. Notably, it performed better than the deterministic strategy when the gesture decoder failed to distinguish commands in similar shapes. For example, the average input error rate for the command "cut" was 34.2% for the deterministic method, because its gesture trace was very similar to "copy" on a Qwerty keyboard. 65% of input for "cut" was misrecognized to "copy" for the deterministic method. BayesianCommand reduced the error rate to 16.2%, showing that combining prior and likelihood resolved some ambiguity introduced in the gesture decoding. BayesianCommand also outperformed the likelihood-only approach by 24.5%. The results also substantiated our claim that fully applying the Bayes' theorem could be adopted in various applications to deal with the input uncertainty.

Likewise, limiting the number of failed attempts to 3 had minor effects on the results. Nine participants correctly finished all trials in under three tries in all conditions. For the rest 9 participants, the average percentage of trials that failed three times was $1.30 \pm 1.67\%$ for the deterministic approach, $1.02 \pm 1.53\%$ for likelihood-only, and $1.30 \pm 2.78\%$ for BayesianCommand. Compared to the other two conditions, BayesianCommand did not introduce more trials that failed three times or contained inaccessible commands. The percentage increased over Experiment I as gesture input is a more complex procedure with higher cognitive and motor execution demands.

While the application focused on word-gesture shortcuts, BayesianCommand could be extended to other gestural command input methods, e.g., Command Strokes [38], CommandBoard [1], or HotStrokes [17]. The prior probability and the likelihood models are independent of the gesture decoder, thus being applicable to other gestural command input methods with minor modification. Investigation on the generalization and other recall-based methods are interesting future work.

LIMITATIONS AND FUTURE WORK

A side effect of incorporating prior probability is that it could make the less frequent items difficult to select. Although our experiments did not show severe consequences, the infrequent items would become more and more challenging to select as their prior probabilities are decreasing [44]. We could mitigate the problem by adding a lower bound for command frequency to ensure that no command will become hard to access or inaccessible. In real-world applications, we could leverage user actions to address them. For example, if the previous selection is an error (back/cancel button is pressed immediately), the probability of this command will decrease for the subsequent command input, preventing users from repeatedly selecting the same incorrect command and increasing the chance of selecting the intended one.

Our investigation on point-and-click input was under the assumption that the target size decided the likelihood model $P(\mathbf{s}|c)$. Such a hypothesis did not reflect the possibility that users may adapt their interaction behavior as frequent items were becoming easier to select. It is worth investigating whether adapting $P(\mathbf{s}|c)$ according to user interaction experience would lead to more accurate likelihood models.

Additionally, BayesianCommand is essentially adjusting the command activation space according to command frequencies. In the current investigation, we did not communicate such adjustment to users via visuals to avoid disruption caused by interface visual changes. It is worth investigating whether communicating this adjustment would affect users' interaction behavior and how we develop more accurate likelihood models to capture it.

Our two examples (i.e., point-and-click and gestural command input) were two experiments demonstrating the effectiveness of BayesianCommand. Many design choices for the models (e.g., the value of k, the command set, and the trigger words) were specific to these experiment settings. As shown in [21], we could make more mature decisions for real-world applications with more contextual information such as the interaction scenarios, command set sizes, and users' preferences. For example, if we can access detailed command input history (e.g., command usage patterns for Microsoft Word [29]), we may build a more advanced prior model (e.g., n-gram command sequence) and follow the similar principle proposed in this paper to improve the command selection accuracy.

CONCLUSIONS

In this paper, we have formalized Bayes' theorem as a guiding framework for deciding the target in command input. To support this principle, we have developed three models: (1) a prior probability model, (2) a dual-Gaussian likelihood model for point-and-click, and (3) a two-step likelihood model for word-gesture shortcuts. Our experiments showed that applying BayesianCommand with the proposed models substantially improved the command input accuracy. Compared to the deterministic principles or applying the principle partially, BayesianCommand reduced the command input error rate by 26.9% and 28.0% for point-and-click, and by 39.9% and 24.5% for word-gesture command input.

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 (UIST '17)*. ACM, New York, NY, USA, 17–28. DOI: http://dx.doi.org/10.1145/3126594.3126639

[2] Georg Apitz, François Guimbretière, and Shumin Zhai. 2008. Foundations for Designing and Evaluating User Interfaces Based on the Crossing Paradigm. *ACM Trans. Comput.-Hum. Interact.* 17, 2, Article 9 (May 2008), 42 pages. DOI:http://dx.doi.org/10.1145/1746259.1746263

[3] Caroline Appert and Shumin Zhai. 2009. Using Strokes As Command Shortcuts: Cognitive Benefits and Toolkit Support. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 2289–2298. DOI: http://dx.doi.org/10.1145/1518701.1519052

[4] Shiri Azenkot and Shumin Zhai. 2012. Touch Behavior with Different Postures on Soft Smartphone Keyboards. In *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services (MobileHCI '12)*. ACM, New York, NY, USA, 251–260. DOI: http://dx.doi.org/10.1145/2371574.2371612

[5] 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 (INTERACT'07)*. Springer-Verlag, Berlin, Heidelberg, 475–488. http://dl.acm.org/citation.cfm?id=1776994.1777053

[6] 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 (AVI '08)*. ACM, New York, NY, USA, 15–22. DOI: http://dx.doi.org/10.1145/1385569.1385575

[7] Olivier Bau and Wendy E. Mackay. 2008. OctoPocus: A Dynamic Guide for Learning Gesture-based Command Sets. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology (UIST '08)*. ACM, New York, NY, USA, 37–46. DOI: http://dx.doi.org/10.1145/1449715.1449724

[8] Hrvoje Benko, Andrew D. Wilson, and Patrick Baudisch. 2006. Precise Selection Techniques for Multi-touch Screens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '06)*. ACM, New York, NY, USA, 1263–1272. DOI: http://dx.doi.org/10.1145/1124772.1124963

[9] Xiaojun Bi, Yang Li, and Shumin Zhai. 2013. FFitts Law: Modeling Finger Touch with Fitts' Law. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 1363–1372. DOI: http://dx.doi.org/10.1145/2470654.2466180

[10] Xiaojun Bi and Shumin Zhai. 2013. Bayesian Touch: A Statistical Criterion of Target Selection with Finger Touch. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST '13)*. ACM, New York, NY, USA, 51–60. DOI: http://dx.doi.org/10.1145/2501988.2502058

[11] Xiaojun Bi and Shumin Zhai. 2016. Predicting Finger-Touch Accuracy Based on the Dual Gaussian Distribution Model. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. ACM, New York, NY, USA, 313–319. DOI: http://dx.doi.org/10.1145/2984511.2984546

[12] Renaud Blanch, Yves Guiard, and Michel Beaudouin-Lafon. 2004. Semantic Pointing: Improving Target Acquisition with Control-display Ratio Adaptation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. ACM, New York, NY, USA, 519–526. DOI: http://dx.doi.org/10.1145/985692.985758

[13] Daniel Buschek and Florian Alt. 2015. TouchML: A Machine Learning Toolkit for Modelling Spatial Touch Targeting Behaviour. In *Proceedings of the 20th International Conference on Intelligent User Interfaces (IUI '15)*. ACM, New York, NY, USA, 110–114. DOI: http://dx.doi.org/10.1145/2678025.2701381

[14] Daniel Buschek and Florian Alt. 2017. ProbUI: Generalising Touch Target Representations to Enable Declarative Gesture Definition for Probabilistic GUIs. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 4640–4653. DOI: http://dx.doi.org/10.1145/3025453.3025502

[15] Eun Kyoung Choe, Kristen Shinohara, Parmit K. Chilana, Morgan Dixon, and Jacob O. Wobbrock. 2009. Exploring the Design of Accessible Goal Crossing Desktop Widgets. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems (CHI EA '09)*. ACM, New York, NY, USA, 3733–3738. DOI: http://dx.doi.org/10.1145/1520340.1520563

[16] 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 (CHI '07)*. ACM, New York, NY, USA, 627–636. DOI: http://dx.doi.org/10.1145/1240624.1240723

[17] 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 (CHI '19)*. ACM, New York, NY, USA, Article 165, 13 pages. DOI: http://dx.doi.org/10.1145/3290605.3300395

[18] Dictionary.com. 2019. Thesaurus.com | Synonyms and Antonyms of Words. https://www.thesaurus.com/. (2019). https://www.thesaurus.com/ [Online; accessed 6-August-2019].

[19] Stephen R. Ellis and Robert J. Hitchcock. 1986. The Emergence of Zipf's Law: Spontaneous Encoding Optimization by Users of a Command Language. *IEEE Trans. Syst. Man Cybern.* 16, 3 (May 1986), 423–427. DOI:http://dx.doi.org/10.1109/TSMC.1986.4308973

[20] Leah Findlater and Krzysztof Z. Gajos. 2009. Design Space and Evaluation Challenges of Adaptive Graphical User Interfaces. *AI Magazine* 30, 4 (Sept. 2009), 68. DOI:http://dx.doi.org/10.1609/aimag.v30i4.2268

[21] Stephen Fitchett and Andy Cockburn. 2012. AccessRank: Predicting What Users Will Do Next. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 2239–2242. DOI: http://dx.doi.org/10.1145/2207676.2208380

[22] Jérémie Francone, Gilles Bailly, Eric Lecolinet, Nadine Mandran, and Laurence Nigay. 2010. Wavelet Menus on Handheld Devices: Stacking Metaphor for Novice Mode and Eyes-free Selection for Expert Mode. In *Proceedings of the International Conference on Advanced Visual Interfaces (AVI '10)*. ACM, New York, NY, USA, 173–180. DOI: http://dx.doi.org/10.1145/1842993.1843025

[23] 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 (MobileHCI '09)*. ACM, New York, NY, USA, Article 49, 4 pages. DOI: http://dx.doi.org/10.1145/1613858.1613919

[24] 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 (CHI '17)*. ACM, New York, NY, USA, 5630–5642. DOI: http://dx.doi.org/10.1145/3025453.3025486

[25] Mayank Goel, Leah Findlater, and Jacob Wobbrock. 2012a. WalkType: Using Accelerometer Data to Accomodate Situational Impairments in Mobile Touch Screen Text Entry. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 2687–2696. DOI:http://dx.doi.org/10.1145/2207676.2208662

[26] Mayank Goel, Jacob Wobbrock, and Shwetak Patel. 2012b. GripSense: Using Built-in Sensors to Detect Hand Posture and Pressure on Commodity Mobile Phones. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST '12)*. ACM, New York, NY, USA, 545–554. DOI: http://dx.doi.org/10.1145/2380116.2380184

[27] Joshua Goodman, Gina Venolia, Keith Steury, and Chauncey Parker. 2002. Language Modeling for Soft Keyboards. In *Proceedings of the 7th International Conference on Intelligent User Interfaces (IUI '02)*. ACM, New York, NY, USA, 194–195. DOI: http://dx.doi.org/10.1145/502716.502753

[28] Tovi Grossman, Pierre Dragicevic, and Ravin Balakrishnan. 2007. Strategies for Accelerating On-line Learning of Hotkeys. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*. ACM, New York, NY, USA, 1591–1600. DOI:http://dx.doi.org/10.1145/1240624.1240865

[29] Jensen Harris. 2006. No Distaste for Paste (Why the UI, Part 7). https://blogs.msdn.microsoft.com/jensenh/2006/04/07/no-distaste-for-paste-why-the-ui-part-7/. (2006). [Online; accessed 19-August-2019].

[30] 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*. Advances in psychology, Vol. 52. North-Holland, Oxford, England, 139–183.

[31] Niels Henze, Enrico Rukzio, and Susanne Boll. 2011. 100,000,000 Taps: Analysis and Improvement of Touch Performance in the Large. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI '11)*. ACM, New York, NY, USA, 133–142. DOI:http://dx.doi.org/10.1145/2037373.2037395

[32] 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 (CHI '10)*. ACM, New York, NY, USA, 581–590. DOI: http://dx.doi.org/10.1145/1753326.1753413

[33] Christian Holz and Patrick Baudisch. 2011. Understanding Touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 2501–2510. DOI:http://dx.doi.org/10.1145/1978942.1979308

[34] Hyun Ka. 2013. *Circling Interface: An Alternative Interaction Method for On-Screen Object Manipulation*. Ph.D. Dissertation. University of Pittsburgh.

[35] Kenrick Kin, Björn Hartmann, Tony DeRose, and Maneesh Agrawala. 2012a. Proton++: A Customizable Declarative Multitouch Framework. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST '12)*. ACM, New York, NY, USA, 477–486. DOI: http://dx.doi.org/10.1145/2380116.2380176

[36] Kenrick Kin, Björn Hartmann, Tony DeRose, and Maneesh Agrawala. 2012b. Proton: Multitouch Gestures As Regular Expressions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 2885–2894. DOI:http://dx.doi.org/10.1145/2207676.2208694

[37] 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 (UIST '04)*. ACM, New York, NY, USA, 43–52. DOI:http://dx.doi.org/10.1145/1029632.1029640

[38] 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 (CHI '07)*. ACM, New York, NY, USA, 1137–1146. DOI: http://dx.doi.org/10.1145/1240624.1240797

[39] Gordon Kurtenbach and William Buxton. 1994. User Learning and Performance with Marking Menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '94)*. ACM, New York, NY, USA, 258–264. DOI: http://dx.doi.org/10.1145/191666.191759

[40] 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.

[41] 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. DOI: http://dx.doi.org/10.1207/s15327051hci0801_1

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

[43] Yang Li. 2010b. Protractor: A Fast and Accurate Gesture Recognizer. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, New York, NY, USA, 2169–2172. DOI:http://dx.doi.org/10.1145/1753326.1753654

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

[45] Wanyu Liu, Rafael Lucas D'Oliveira, Michel Beaudouin-Lafon, and Olivier Rioul. 2017b. BIGnav: Bayesian Information Gain for Guiding Multiscale Navigation. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 5869–5880. DOI: http://dx.doi.org/10.1145/3025453.3025524

[46] Wanyu Liu, Olivier Rioul, Joanna McGrenere, Wendy E. Mackay, and Michel Beaudouin-Lafon. 2018. BIGFile: Bayesian Information Gain for Fast File Retrieval. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 385, 13 pages. DOI: http://dx.doi.org/10.1145/3173574.3173959

[47] Hao Lü, James A. Fogarty, and Yang Li. 2014. Gesture Script: Recognizing Gestures and Their Structure Using Rendering Scripts and Interactively Trained Parts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 1685–1694. DOI: http://dx.doi.org/10.1145/2556288.2557263

[48] 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 (CHI '11)*. ACM, New York, NY, USA, 207–216. DOI: http://dx.doi.org/10.1145/1978942.1978972

[49] 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 (CHI '13)*. ACM, New York, NY, USA, 257–266. DOI: http://dx.doi.org/10.1145/2470654.2470690

[50] 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 (CHI '15)*. ACM, New York, NY, USA, 3355–3364. DOI: http://dx.doi.org/10.1145/2702123.2702610

[51] Yuexing Luo and Daniel Vogel. 2014. Crossing-based Selection with Direct Touch Input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 2627–2636. DOI: http://dx.doi.org/10.1145/2556288.2557397

[52] Yuexing Luo and Daniel Vogel. 2015. Pin-and-Cross: A Unimanual Multitouch Technique Combining Static Touches with Crossing Selection. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15)*. ACM, New York, NY, USA, 323–332. DOI: http://dx.doi.org/10.1145/2807442.2807444

[53] 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 (CHI '18)*. ACM, New York, NY, USA, Article 344, 13 pages. DOI: http://dx.doi.org/10.1145/3173574.3173918

[54] Tomer Moscovich. 2009. Contact Area Interaction with Sliding Widgets. In *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology (UIST '09)*. ACM, New York, NY, USA, 13–22. DOI:http://dx.doi.org/10.1145/1622176.1622181

[55] Josip Musić and Roderick Murray-Smith. 2016. Nomadic Input on Mobile Devices: The Influence of Touch Input Technique and Walking Speed on Performance and Offset Modeling. *Human-Computer Interaction* 31, 5 (Sept. 2016), 420–471. DOI: http://dx.doi.org/10.1080/07370024.2015.1071195

[56] Douglas L Nelson, Valerie S Reed, and John R Walling. 1976. Pictorial superiority effect. *Journal of Experimental Psychology: Human Learning and Memory* 2, 5 (1976), 523.

[57] Alex Olwal, Steven Feiner, and Susanna Heyman. 2008. Rubbing and Tapping for Precise and Rapid Selection on Touch-screen Displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, New York, NY, USA, 295–304. DOI: http://dx.doi.org/10.1145/1357054.1357105

[58] Charles Perin, Pierre Dragicevic, and Jean-Daniel Fekete. 2015. Crossets: Manipulating Multiple Sliders by Crossing. In *Proceedings of the 41st Graphics Interface Conference (GI '15)*. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 233–240. http://dl.acm.org/citation.cfm?id=2788890.2788931

[59] Simon Rogers, John Williamson, Craig Stewart, and Roderick Murray-Smith. 2010. FingerCloud: Uncertainty and Autonomy Handover Incapacitive Sensing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, New York, NY, USA, 577–580. DOI: http://dx.doi.org/10.1145/1753326.1753412

[60] Anne Roudaut, Eric Lecolinet, and Yves Guiard. 2009. MicroRolls: Expanding Touch-screen Input Vocabulary by Distinguishing Rolls vs. Slides of the Thumb. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 927–936. DOI: http://dx.doi.org/10.1145/1518701.1518843

[61] Julia Schwarz, Scott Hudson, Jennifer Mankoff, and Andrew D. Wilson. 2010. A Framework for Robust and Flexible Handling of Inputs with Uncertainty. In *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology (UIST '10)*. ACM, New York, NY, USA, 47–56. DOI: http://dx.doi.org/10.1145/1866029.1866039

[62] Julia Schwarz, Jennifer Mankoff, and Scott Hudson. 2011. Monte Carlo Methods for Managing Interactive State, Action and Feedback Under Uncertainty. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, New York, NY, USA, 235–244. DOI: http://dx.doi.org/10.1145/2047196.2047227

[63] Daniel Vogel and Ravin Balakrishnan. 2010. Occlusion-aware Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, New York, NY, USA, 263–272. DOI: http://dx.doi.org/10.1145/1753326.1753365

[64] Daniel Vogel and Patrick Baudisch. 2007. Shift: A Technique for Operating Pen-based Interfaces Using Touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*. ACM, New York, NY, USA, 657–666. DOI: http://dx.doi.org/10.1145/1240624.1240727

[65] Daniel Vogel and Géry Casiez. 2012. Hand Occlusion on a Multi-touch Tabletop. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 2307–2316. DOI:http://dx.doi.org/10.1145/2207676.2208390

[66] David J. Ward, Alan F. Blackwell, and David J. C. MacKay. 2000. Dasher-a Data Entry Interface Using Continuous Gestures and Language Models. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology (UIST '00)*. ACM, New York, NY, USA, 129–137. DOI: http://dx.doi.org/10.1145/354401.354427

[67] Daryl Weir, Simon Rogers, Roderick Murray-Smith, and Markus Löchtefeld. 2012. A User-specific Machine Learning Approach for Improving Touch Accuracy on Mobile Devices. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST '12)*. ACM, New York, NY, USA, 465–476. DOI: http://dx.doi.org/10.1145/2380116.2380175

[68] 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 (UIST '07)*. ACM, New York, NY, USA, 269–278. DOI: http://dx.doi.org/10.1145/1294211.1294259

[69] 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 (UIST '06)*. ACM, New York, NY, USA, 259–268. DOI: http://dx.doi.org/10.1145/1166253.1166294

[70] John Williamson. 2006. *Continuous Uncertain Interaction*. Ph.D. Dissertation. University of Glasgow.

[71] Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. 2007. Gestures Without Libraries, Toolkits or Training: A $1 Recognizer for User Interface Prototypes. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology (UIST '07)*. ACM, New York, NY, USA, 159–168. DOI: http://dx.doi.org/10.1145/1294211.1294238

[72] Wenchang Xu, Chun Yu, and Yuanchun Shi. 2011. RegionalSliding: Enhancing Target Selection on Touchscreen-based Mobile Devices. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems (CHI EA '11)*. ACM, New York, NY, USA, 1261–1266. DOI: http://dx.doi.org/10.1145/1979742.1979758

[73] Koji Yatani, Kurt Partridge, Marshall Bern, and Mark W. Newman. 2008. Escape: A Target Selection Technique Using Visually-cued Gestures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, New York, NY, USA, 285–294. DOI: http://dx.doi.org/10.1145/1357054.1357104

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

[75] 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 (UIST '04)*. ACM, New York, NY, USA, 33–42. DOI: http://dx.doi.org/10.1145/1029632.1029639

[76] 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 (CHI '18)*. ACM, New York, NY, USA, Article 249, 14 pages. DOI: http://dx.doi.org/10.1145/3173574.3173823

[77] 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 (CHI '18)*. ACM, New York, NY, USA, Article 439, 13 pages. DOI: http://dx.doi.org/10.1145/3173574.3174013

[78] 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 (CHI '19)*. ACM, New York, NY, USA, Article 448, 12 pages. DOI: http://dx.doi.org/10.1145/3290605.3300678

[79] George Kingsley Zipf. 1949. *Human behavior and the principle of least effort: an introduction to human ecology*. Addison-Wesley Press.

APPENDIX

TRIGGER WORDS OF THE COMMANDS

Table 4 shows the 11 trigger words of the 20 commands in Experiment II. The words in bold were the trigger words used by the participants to trigger the corresponding commands in the deterministic condition. Note that these words represent the decoding output from the gesture recognizer, not necessarily what the participants intended to input.

Command	Trigger words
	calculator, calculators, calculate, calculation, compute, computer, computation, microcomputer, count, appraise, spreadsheet
	**camera**, **cameras**, camcorder, video, photograph, photographer, **cameraman**, videocamera, tripod, lens, projector
	clock, clocks, timer, time, dial, watch, stopwatch, alarm, tick, seconds, wristwatch
	**copy**, copying, copyist, replicate, replica, imitate, reproduce, emulate, duplicate, plagiarize, clone
	**cut**, cutting, slice, trim, reduce, prune, shorten, truncate, curtail, scissors, clippers
	**delete**, deleting, deleted, **deletes**, deletion, remove, uninstall, eliminate, omit, overwrite, discard
	**download**, **downloads**, downloadable, upload, redownload, load, **downloader**, browse, access, **file-sharing**, homepage
	**edit**, **editing**, editor, **edits**, **edited**, annotate, annotated, **essay**, alter, revise, rewrite
	**file**, files, **filing**, filename, filed, **refile**, **folder**, document, documents, archive, directory
	help, helping, helps, helped, assist, assistance, aid, support, avail, advice, service
	**keyboard**, **keyboards**, touchpad, trackpad, keypad, qwerty, stylus, numberpad, typewriter, typing, laptop
	**mail**, mails, **mailbox**, mailing, e-mail, email, spam, letter, postal, post, **mailed**
	network, networks, networked, net, internet, web, cable, channel, connectivity, networking, interconnect
	**print**, **printing**, **printer**, printed, reprint, handwritten, **photocopy**, **publish**, publication, booklet, distribute
	recent, subsequent, recently, latest, previous, past, earlier, prior, preceding, later, coming
	**rotate**, rotation, **rotational**, tilted, pivot, tilt, rotating, **rotated**, revolving, swivel, spin
	**search**, searches, **searching**, retrieve, discover, check, find, look, quest, searcher, scour
	share, shared, sharing, exchange, swap, commonality, pool, combine, express, collect, common
	weather, inclement, meteorological, windy, forecast, forecaster, winter, foggy, thunderstorm, meteorologist, blizzard
	zoom, zoom-in, close-up, enlarge, magnify, magnifier, scroll, augment, enhance, expand, amplify

Table 4. The trigger words for the 20 commands used in Experiment II.