TOAST: Ten-Finger Eyes-Free Typing on Touchable Surfaces

WEINAN SHI, CHUN YU*, XIN YI, ZHEN LI, and YUANCHUN SHI, Tsinghua University, China

Touch typing on flat surfaces (e.g. interactive tabletop) is challenging due to lack of tactile feedback and hand drifting. In this paper, we present TOAST, an eyes-free keyboard technique for enabling efficient touch typing on touch-sensitive surfaces. We first formalized the problem of keyboard parameter (e.g. location and size) estimation based on users' typing data. Through a user study, we then examined users' eyes-free touch typing behavior on an interactive tabletop with only asterisk feedback. We fitted the keyboard model to the typing data, results suggested that the model parameters (keyboard location and size) changed not only between different users, but also within the same user along with time. Based on the results, we proposed a Markov-Bayesian algorithm for input prediction, which considers the relative location between successive touch points within each hand respectively. Simulation results showed that based on the pooled data from all users, this model improved the top-1 accuracy of the classical statistical decoding algorithm from 86.2% to 92.1%. In a second user study, we further improved TOAST with dynamical model parameter adaptation, and evaluated users' text entry performance with TOAST using realistic text entry tasks. Participants reached a pick-up speed of 41.4 WPM with a character-level error rate of 0.6%. And with less than 10 minutes of practice, they reached 44.6 WPM without sacrificing accuracy. Participants' subjective feedback also indicated that TOAST offered a natural and efficient typing experience.

CCS Concepts: • **Human-centered computing** → **Text input**; *Ubiquitous computing*;

Additional Key Words and Phrases: Text entry, ten-finger, eyes-free, adaptive keyboard

ACM Reference Format:
Weinan Shi, Chun Yu, Xin Yi, Zhen Li, and Yuanchun Shi. 2018. TOAST: Ten-Finger Eyes-Free Typing on Touchable Surfaces. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 1, Article 33 (March 2018), 23 pages. https://doi.org/10.1145/3191765

1 INTRODUCTION

With the popularity of ubiquitous computing environments (e.g. interactive surfaces, augmented reality, tabletops), the need for text entry under these scenarios has increased significantly. However, the challenge of providing text entry experience with high efficiency still bothers users and researchers. Among possible alternatives, ten-finger typing on flat surfaces offers a natural and potentially most efficient text entry experience, as users are able to transfer their muscle memory on physical keyboards to this scenario [9].

Despite this potential, there are several challenges that hinder the performance of touch typing on flat surfaces: 1) In ubiquitous text entry scenarios, the information output interface is usually decoupled from the input interface (e.g. smart TVs and virtual reality). As a result, there would be no keyboard for users to align with, which makes it ambiguous when interpreting the touch location; 2) During touch typing, users' visual attention

*Corresponding author

Authors' addresses: Weinan Shi, swn16@mails.tsinghua.edu.cn; Chun Yu, chunyu@tsinghua.edu.cn; Xin Yi, yix15@mails.tsinghua.edu.cn; Zhen Li, lizhen11@mails.tsinghua.edu.cn; Yuanchun Shi, shiyc@tsinghua.edu.cn, Department of Computer Science and Technology, Global Innovation eXchange Institute, Beijing National Research Center for Information Science and Technology, Tsinghua University, Haidian District, Beijing, 100084, China; Key Laboratory of Pervasive Computing, Ministry of Education, China.

2474-9567/2018/3-ART33 $15.00
https://doi.org/10.1145/3191765

is usually not focused on the keyboard layout, which could lead to imprecision in touch location [9]; 3) when typing on flat surfaces, the lack of tactile feedback would lead to hand drifting during typing [16].

Aimed at these challenges, a number of researches have been conducted to understand users' ten-finger typing behavior [4, 9], to support tactile feedback [33], or to develop ten-finger keyboard techniques that leveraged different sensors [4, 14, 23, 25]. However, we see two major limitations in existing works: 1) When examining the consistency of users' typing behavior over time, existing works only focus on the location of the keyboard (usually referred to as *hand drifting* [16]), which limits the analysis of more keyboard parameters. For example, as we will show in this paper, the imaginary keyboard size of users also changes across time in both horizontal and vertical dimensions; 2) Despite the great success of statistical decoding algorithms (e.g. [11]) in mobile text entry techniques (e.g. [5, 31]), to our knowledge, no one has explored the feasibility of incorporating such algorithms in ten-finger typing on flat surface scenario. As we will show, with a Markov-Bayesian algorithm and online adaptation feature, users could reach a 11-44% higher text entry speed than with existing techniques.

In this paper, we adopt an iterative approach in supporting ten-finger eyes-free typing on flat surfaces. First, we formalized the problem of keyboard parameter estimation based on users' typing data, which accounts for different sources of variance (e.g. keyboard location, size and shape). To understand users' ten-finger eyes-free typing pattern, we conducted a user study with only asterisk feedback. We fitted the keyboard model to the collected typing data, and found that model drifting not only happened on keyboard location, but also on keyboard shape. Moreover, the model parameters changed not only among different users, but also within the same user over time.

Based on the observations, we proposed TOAST, a keyboard technique that enables ten-finger eyes-free typing on flat surface based on the reported touch point location. The algorithm of TOAST has two essential features: 1) To effectively mitigate the variation in the keyboard location, TOAST predicts user input based on the relative location between successive touch points within two hands respectively, which extends the classical statistical decoding algorithm that considers each touch point independently [11]; 2) To account for the model drifting during typing, TOAST leverages an adaptive mechanism to dynamically adjust the estimated keyboard location and size. Compared with key-press models (e.g. [8]), this approach requires less training data, and are potentially more tolerable to input errors.

We verified the performance of TOAST through simulation, where we compared the word-level prediction accuracy of different algorithms using the user data in the first user study. Results showed that the proposed Markov-Bayesian algorithm yielded significant higher accuracy than the classical statistical decoding algorithm on the pooled data, while the two algorithms yielded competitive performance on user-specific data. In a further evaluation study with real text entry tasks, we found that within 10 minutes of practicing, users could achieve 45 WPM with negligible error rates, with expert users even reaching 66.1 WPM. Subjective feedback also indicated that participants felt TOAST easy to learn and offers a fast and accurate text entry experience.

Specifically, our contributions in this paper are three-fold:

(1) We achieve a more detailed understanding of user's typing behavior (e.g. the change of the emerged keyboards' shape against time) on flat surface than previous researches (e.g. [9, 16]) when conducting ten-finger eyes-free typing and with no extra calibration for users during typing.
(2) We augment the classical statistical decoding algorithm by using relative information between touch points, considering left and right hand keyboard model respectively, and adopting an adaptive strategy to develop an eyes-free ten-finger typing technique.
(3) We are the first to evaluate the feasibility and performance of eyes-free ten-finger typing on touch sensitive flat surface with an statistical decoding algorithm.

Though in this paper, we focused on the design of input prediction algorithm based on the reported touch points of a touchable surface, we believe that our algorithms can be incorporated with various touch sensing

techniques (e.g. through computer vision in [23] or infrared light in [25]). And by so, our results in this paper may also be applied to a wider range of scenarios, and push the goal towards touch typing on any surface a step further.

In the remaining of this paper, after the introduction of related work, we first formally describe the problem of keyboard model estimation based on users' typing data, which serves as a basis of TOAST. We then report our study that examined users' ten-finger eyes-free typing behavior, followed by the description of the algorithm of TOAST. Finally, we report the simulation and evaluation study results, which verified the effectiveness of TOAST on supporting ten-finger eyes-free typing on flat surface. We believe that our results in this paper is insightful for understanding users' text entry behavior on flat surfaces, and our technique approach advances the state-of-the-art of text entry to more ubiquitous scenarios.

2 RELATED WORK

We will review related works about the understanding of human's ten-finger typing , input techniques enabling typing by ten-fingers, statistical decoding algorithms used in text entry and adaptive keyboards.

2.1 Understanding Ten-Finger Typing Behavior

Ten-finger typing on a physical keyboard is a very common means for inputting text [19, 30], where users can reach a type speed of 60-100 words per minute. Usually, ten-finger typing can be achieved by using the muscle memory and the tactile feedback provided by the physical keys, which leads to an eyes-free process (e.g. touch typing[1]). However, it is difficult for users to transfer their typing experience from physical keyboard to soft keyboard due to the limited tactile feedback. Researchers [22] showed that touch-typing on software QWERTY keyboards could only achieve 20 WPM, which was much slower than the speed on a physical keyboard, and was close to the speed of single-finger typing (e.g. with one index or thumb).

However, few researches have been conducted to investigate the intrinsic behavior of ten-finger typing. Findlater et al. [9] examined typing patterns (e.g. how hands and fingers were placed) of expert users when performing ten-finger typing on a flat surface. The distribution of touch points under conditions of a visible keyboard and an visible keyboard were investigated. As a result, they concluded that eyes-free ten-finger typing could be possible. However, users were required to register their hand location before each trail, which aimed at reducing the effects of hand drift, thus limited the continuous understanding of typing process. Li et al. [16] studied the effect of hand drift while ten-finger typing. They found that hand drift was stronger for invisible keyboards than visible keyboards. They also analyzed the trend of hand drift against time, yet to our knowledge, the change of other keyboard parameters such as keyboard size have not been studied before. We will provide a more detailed understanding towards continuous ten-finger typing behavior in this paper.

2.2 Ten-Finger Typing Techniques

Some typing techniques have been proposed to support ten-finger typing on flat surfaces. TapBoard [14] was a soft keyboard that regarded tapping actions as keystrokes and other touches as the touched stated, which allowed users to rest on the touchscreen and lead to a "tapping to type" mechanism. TapBoard supported ten-finger typing and the reported typing speed was 38.68 WPM. Choi et al. [4] analyzed the correlation between touching fingers and non-touching fingers movement when ten-finger typing. They then proposed a virtual keyboard which outperforms existing ones when all finger positions were known. However, to our knowledge, there have not been any techniques specifically aiming at an eyes-free typing scenario, as well as the typing and correction algorithms.

[1] https://en.wikipedia.org/wiki/Touch_typing

Some of the techniques focused on how to detect and recognize the appropriate finger when performing typing, such as by emitting a plane of infrared light slightly above the typing surface [25] or using a camera and applying machine learning method [23]. Unfortunately, typing behavior and typing algorithms were not considered in these researches.

Other techniques provided different forms of typing by ten-finger, such as typing in the air [6, 36], typing on a condensed 1-line keyboard [17], typing by chording keymaps [26], typing by pressing the thumb against specific contact points [15], typing on both front and back side of a mobile device [28]. However, ten-finger typing on a flat surface using a standard QWERTY layout, which can be transfer directly from typing on a physical keyboard, can be a more common, promising and well-adapted approach. In this paper, we will mainly focus on this specific scenario.

2.3 Statistical Decoding Algorithm

A classical statistical decoding algorithm uses a probabilistic model that calculates each word in a pre-defined dictionary according to user's input, and recommends the most likely word(s). This method was first proposed by Goodman et al. [11], which will be described in detail in Section 5.1. This statistical decoding algorithm has been proved effective in many smart keyboard scenarios, such as on a smart watch [12, 35], on a mobile phone [31, 32], on a tabletop[8], even typing in the air [36]. Yet to our knowledge, the feasibility of this algorithm under the eyes-free ten-finger typing scenario, under which many different variations (e.g. the effects of hand drifting, the difference of keyboard under two hands) need to be considered, has not been explored.

A relative keyboard [24] was developed to deal with situation when the system was agnostic about the location of the keyboard. The idea was to interpret users' input based on the vectors from subsequent hit points to the first one. The authors tested the algorithm's performance with off-line typing data. However, the prediction accuracy was quite low (48%). This could be attributed to the lack of a deep understanding of users' eyes-free typing behavior. Lu et al. [18] explored BlindType, an eyes-free typing technique on a touchpad using one thumb, by using absolute and relative algorithms and got satisfying performance of 17-23 WPM. Inspired by these algorithms, we developed our own augmented decoding algorithm to accommodate eyes-free ten-finger typing on flat surfaces.

2.4 Adaptive Keyboard

Due to the noise generated by users on soft keyboards, researchers have developed a number of adaptive keyboards to follow the change of users' behavior. However, adaptive algorithms have not been explored under eyes-free ten-finger typing scenario. The adaptive mechanism also gives us inspiration of dealing with noise when ten-finger eyes-free typing on flat surface.

CATKey [10] visually adapted the keyboard layout to the spatial model. LiquidKeyboard [27] could adapt to the user's finger positions on a touchscreen, which allowed ten-finger typing as well. Gunawardana et al. [13] identified problems with aggressive adaptation, and constrained the adaptation of keyboard layout to provide more usability. Moreover, the adaptation can also be on the underlined models. Yin et al. [37] proposed a hierarchical approach that allowed a software keyboard adapted to key, postures and individuals. Buschek et al. [3] proposed a clustering-based approach for improving accuracy in back-of-device touch-typing with multiple fingers. However, these techniques were not suitable for eyes-free condition.

Findlater et al. [8] proposed a personalized keyboard for character-level input on a touchable surface with visible keyboards. Their machine-learning algorithm leveraged shape and location features of contact points to predict input. The keyboard adapted the underlying key-press classification models during typing. Evaluation results showed the input speed could reach 28 WPM. However, the utilization of shape information of contact

points limited the external validity of their research since most touch sensing platforms do not support this feature.

3 THE PROBLEM OF KEYBOARD MODEL FITTING

In this paper, we focus on the scenario of ten-finger typing on flat surfaces. However, we believe that the technique has a potential of being used in more ubiquitous situations, by assuming there exists some sensing technique in the future, which can report the location of touch points on the surface. Thus, ten-finger typing can happen on a table, a wall, a window or any flat surface that a finger can tap. When input, the user puts his/her hands out on the surface, and type as if there is a virtual keyboard under the hands (but not shown on the surface). The user can start typing on any location on the surface according to his/her convenience, and can walk around and come back to input anytime he/she wants.

The above descriptions define the unique characteristic of ten-finger typing on flat surface: a user types with limited tactile feedback (i.e. no physical keys), with no visual keyboard, and with no constraints of the keyboard location. All of these impose difficult challenges to realize accurate and efficient typing experience. In this section, we will formulate the problem of keyboard parameter estimation based on users' typing data, which accounts for different sources of variance, as well as the simplification and solution to the problem. At last, we will discuss the comparison of our model and a classic model (key-press model).

3.1 Keyboard Model

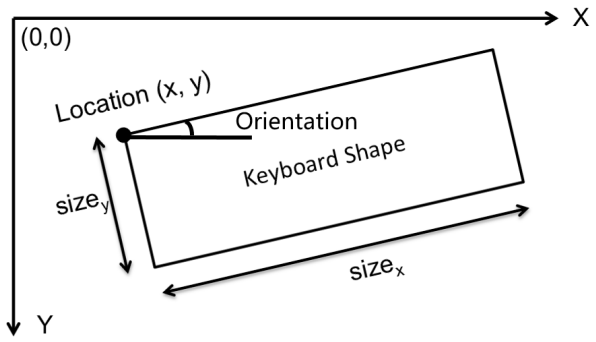


Fig. 1. Illustrating the model of a user keyboard

We assume at any moment, there is a keyboard for input in user's mental model. The user programs and performs touch actions (moving and taping a finger) according to that keyboard. For clarity, we refer the keyboard in user's mental model as a "user keyboard" (or keyboard for simplicity) in this research. We also define a standard keyboard as the absolute layout of a physical keyboard. That is, each key is of a square shape; key distance (measured by the centroid of the key) is 0.75" (or 1.905cm) [2].

The geometry model of a user keyboard can be described by four independent variables (Figure 1):

- **Location**: the coordinate of the user keyboard on the surface. It can be defined using any fixed point on the keyboard. For example, the centroid of the key "Q".

[2] https://en.wikipedia.org/wiki/Keyboard_layout

- **Size**: the width (X-axis size) and height (Y-axis size) of the user keyboard. For example, X-axis size can be measured by the most left key and most right key on the keyboard.
- **Orientation**: the degree of rotation along the X-axis.
- **Shape**: the relative location of keys after the user keyboard is normalized to a standard keyboard. An ideal user keyboard should be rectangle, the same as the standard keyboard.

The four variables describing a user keyboard can be determined and influenced by many factors in terms of typing on flat surface. When the user approaches the surface and puts his/her hands out to input, the location and orientation of the keyboard are determined. During typing, hand drift [16] also changes the location and orientation. In addition, due to the absence of a visual keyboard, keyboard size and shape may dynamically change between typing sessions. Finally, individual users should have different keyboard models, which may depend on their typing habits, hand sizes and finger lengths.

We assume the user's touch actions are programmed toward the centroid of the target keys. However, input noise (n), attributed to human's motor control system, usually occurs during performing the actions. Hence, users' touch points during typing could be generated and influenced by the four keyboard variables and input noise.

3.2 Simplification of the Problem

We make some assumptions to simply the problem for this research.

A1 We assume the user keyboard is always horizontally placed. We do not deal with orientation in this research.

A2 We assume keyboard shape is rectangle, like a standard keyboard that each key has the same size. Researches [9] found the shape could be arc due to the ergonomic structure of human hands. We will provide evidence for making this decision later.

A3 We assume the input noise due to human motor control system is normal [2], and the standard deviation is the same for each individual key.

Based on the simplification, there are three major variables in the typing system: Location (x, y), Key Size (s_x, s_y) and Noise (n_x, n_y). We can thus formulate the actual coordination (x_t, y_t) of an observed touch point as Equation 1, where k_x and k_y denote the location of the key within the QWERTY layout, counted by the number of keys.

$$\begin{pmatrix} x_t \\ y_t \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \begin{pmatrix} k_x \\ k_y \end{pmatrix} + \begin{pmatrix} n_x \\ n_y \end{pmatrix} \tag{1}$$

3.3 Estimation of the Keyboard Model Parameters

A user keyboard describes the keyboard model within users' mental model. So, take X-axis for example, if we have observed a number of touch points $(x_1 \cdots x_n)$ as well as the corresponding (labeled) keys $(c_1 \cdots c_n)$, how can we derive the user keyboard model? We achieve this by estimating the most probable Location (x, y) and Key Size (s_x, s_y) that generate the observed touch points. The maximum likelihood estimation (MLE) can be formulated as Equation 2.

$$(\hat{x}, \hat{s_x}) = \arg\max_{(x, s_x)} P(x_1 \cdots x_n | c_1 \cdots c_n) \tag{2}$$

As most previous researches did (e.g. [11]), we assume independence between each touch point. Then, Equation 2 can be rewritten as Equation 3.

$$(\hat{x}, \hat{s_x}) = \arg\max_{(x, s_x)} \prod_{i=1}^{n} P(x_i | c_i) \tag{3}$$

According to our keyboard model, the touch points of each key should form a Gaussian distribution centered (μ) at the key centroid, with the standard deviation (σ) corresponding to the input noise.

$$(\hat{x}, \hat{s_x}) = \arg\max_{(x, s_x)} \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(x_i - \mu_i)^2}{2\sigma^2}) \tag{4}$$

Since σ is a constant not dependent on the keyboard model, by applying logarithm, we have

$$(\hat{x}, \hat{s_x}) = \arg\min_{(x, s_x)} \sum_{i=1}^{n} (x_i - \mu_i)^2 \tag{5}$$

That is,

$$(\hat{x}, \hat{s_x}) = \arg\min_{(x, s_x)} \sum_{i=1}^{n} (x_i - x - s_x \cdot k_{xi})^2 \tag{6}$$

As shown by Equation 6, the estimation of keyboard model parameters (x, y, s_x, s_y) turns out to be solving a least squares fitting problem. The goodness of model fitting (the R^2 value) indicates how well the shape of user keyboard is conformed to a standard keyboard. An R^2 value of 1 means the user keyboard is absolutely rectangle (not arc). In addition, the residual can be used to estimate the level of input noise (n).

3.4 Keyboard Model versus Key-Press Model

Most of previous researches used the key-press model to describe the user keyboard emerging from observed touch points [8, 9, 11]. A key-press model describes the distribution of touch points for each key separately; it has no explicit consideration about the correlation between keys and their touch points. In contrast, a keyboard model has a uniform representation of the user keyboard (as shown in Equation 1). A keyboard model can imply a key-press model since knowing the keyboard then knowing the keys.

One obvious difference between a keyboard model and a key-press model is the number of parameters required to describe the model. For example, the simplified keyboard model needs only 6 parameters. However, a key-press model needs 4×26=104 parameters (26 bivariate Gaussian distributions). As a result, a key-press model is more expressive: it reveals the detailed shape information of the keyboard. On the other hand, large quantity of typing data is need to train the model, which presents an awkward if the system needs to dynamically adapt to the user's typing behavior quickly. Specifically, if there is no training data for some keys, future touch points on these keys cannot be predicted appropriately. In addition, a key-press model is more sensitive to the training data because the updating of a key-press model is performed directly on individual keys. This can be a problem when the input has much noise. In comparison, updating a keyboard model according to new touch points is more noise proofing.

4 EXAMINING TEN-FINGER TYPING BEHAVIOR

In this section, we carried out an experiment to investigate the typing pattern that emerged when users touch typed on a flat surface without visually attending to a keyboard layout. Our experiment setup was similar as Findlater et al.'s [9]. In the scenario of typing on flat surface, this supposes the user has only one start input. Our goal is to examine the behavior after the start input. However, we are interested in not only the endpoint distribution, but also the differences across participants and the change of the underlying keyboard models over time. In order to observe the very intrinsic typing behavior, we provide only asterisk feedback during typing [2, 9]. The results here are thus indicative of the performance that expert typists were able to achieve when performing ten-finger typing.

4.1 Participants

We recruited 15 participants from the campus (10 male, 5 female), with an average age of 22 (SD = 2.7). All of them used a QWERTY keyboard regularly. We evaluated the expertise of their typing ability on a physical keyboard using TextTest [34]. They typed an average of 55.5 WPM (SD = 19.3) with an uncorrected error rate [29] of 0.2% (SD = 0.8%). Each participant was paid $20.

4.2 Apparatus

In order to simulate the flat surface scenario, we built our experiment platform on a Microsoft Surface 2 (Figure 2a). It has a 35×20 inches multi-touch display with a resolution of 1920×1080. The touchscreen senses multi-touch taps and gestures using infrared rays. The system API reports the x and y coordinates in pixels (1 pixel=0.46 mm) of each touch point with millisecond timestamps. Note that we use Microsoft Surface 2 for its sensing ability and the big enough interaction area. Also, we chose to sense touch and display information at the same surface for the convenience of controlling experiment conditions.

(a) (b)

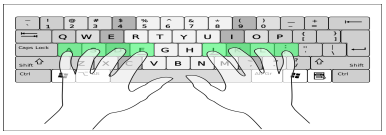
Fig. 2. (a)Experiment setup. (b)The gloves used in the experiment.

During the experiment, participants wore a pair of black gloves (Figure 2b) with the fingertips exposed. Microsoft Surface 2 senses objects on the touchscreen by infrared reflection. When typing, the palm sometimes triggers an unexpected touch event even if it does not have contact with the surface. We observed that false detection was likely to happen when the participants typed fast. We believe this issue should not be present with other sensing techniques (e.g. a capacitive touchscreen). The gloves were used to deal with this recognition problem. The gloves were chosen to be very thin and light, and we confirmed with the users that the gloves did not affect their typing behavior.

4.3 Typing Interface

Figure 3 shows the platform interface. The task phrases were shown on the top, and a rectangular touch area was placed in the center for text input. During typing, participants could rest their palms on the screen outside the touch area to minimize fatigue. Also, taps outside the touch area were ignored and would not trigger feedback.

In order to help participants to familiarize themselves with ten-finger typing on the touchscreen, the platform showed a QWERTY keyboard during warming up sessions. The size of the keyboard was the same as a standard physical keyboard (with a 0.75” key size). During the test sessions, the keyboard would be hidden in order to mimic the scenario in which users typed on flat surfaces without visual feedbacks. During typing, an asterisk



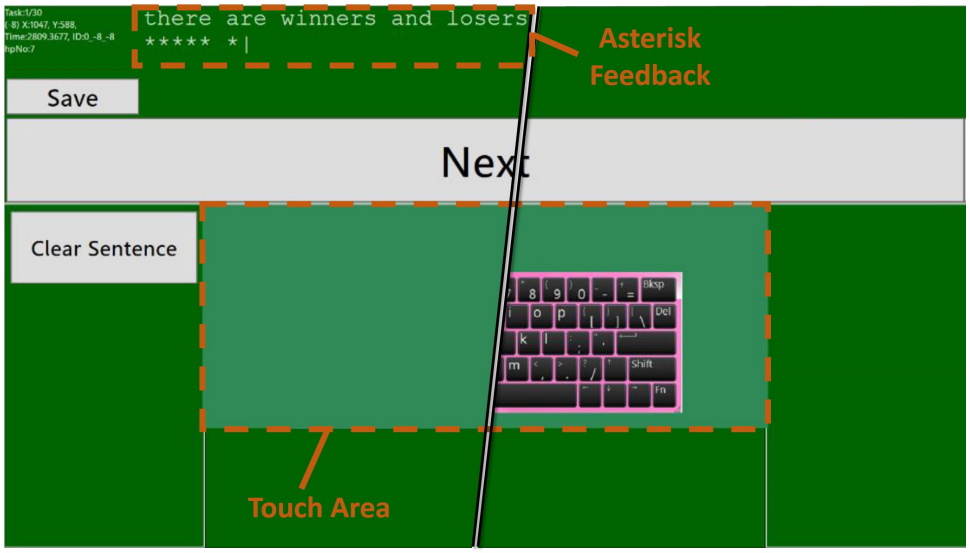


Fig. 3. Platform interface. During warming up, a keyboard layout is showed(right). During typing, the keyboard is hidden and it only shows asterisk feedback(left).

would appear each time a legal tap was detected. This is necessary for ensuring a 1:1 mapping between the logged taps and the input characters, while avoiding the bias towards any specific tap detection algorithm [2, 9].

4.4 Procedure

Before the experiment, we first collected the demographic information of participants using questionnaires. During the experiment, a participant sat before the Surface table, and performed ten-finger typing tasks (see Figure 2a). Before testing, the participant was first allowed five minutes to familiarize himself/herself with ten-finger typing on a visual keyboard. During the test session, the keyboard was turned off. We described there was an imaginary keyboard on the surface as large as a standard keyboard. Before typing, the participant first pressed his/her fingers on the surface for one second to register the imaginary keyboard's location, with eight fingers on the home row and two thumbs on the space. The participant then transcribed 30 phrases that were randomly sampled from the Mackenzie and Soukoreff phrase set [21]. We ensured that all twenty six letters were covered. Participants were asked to type "as fast and as accurately as possible", and to restart a trial if he/she felt a typing error was committed.

4.5 Results

We collected 12,669 taps from all participants in total, not including those in warm-up sessions. To account for mislabeling, for each user, we removed outliers for each key that were more than 3 standard deviations away from the mean of the endpoints in either x or y dimensions (478 / 12,669 = 3.8% of the total taps were removed).

4.5.1 Text Entry Speed. We calculated the text entry speed using the formula in [20]:

$$WPM = \frac{|L| - 1}{T} \times 60 \times \frac{1}{5} \tag{7}$$

where L is the length of the final transcribed string, and T is the time elapse (in seconds) from the first tap to the last tap in the sentence. Participants typed at an average speed of 43.2 WPM (SD = 13.8), which was 22% slower than the speed on the physical keyboard. This was consistent with Findlater et al.'s result [9] that typing on a software keyboard was slower than on a physical keyboard.

4.5.2 Touch Point Distribution. We aligned the observed centroids of "F" and "J" keys from different participants [9] to account for individual difference on the keyboard location. Figure 4 shows the collective touch points merged across all participants.

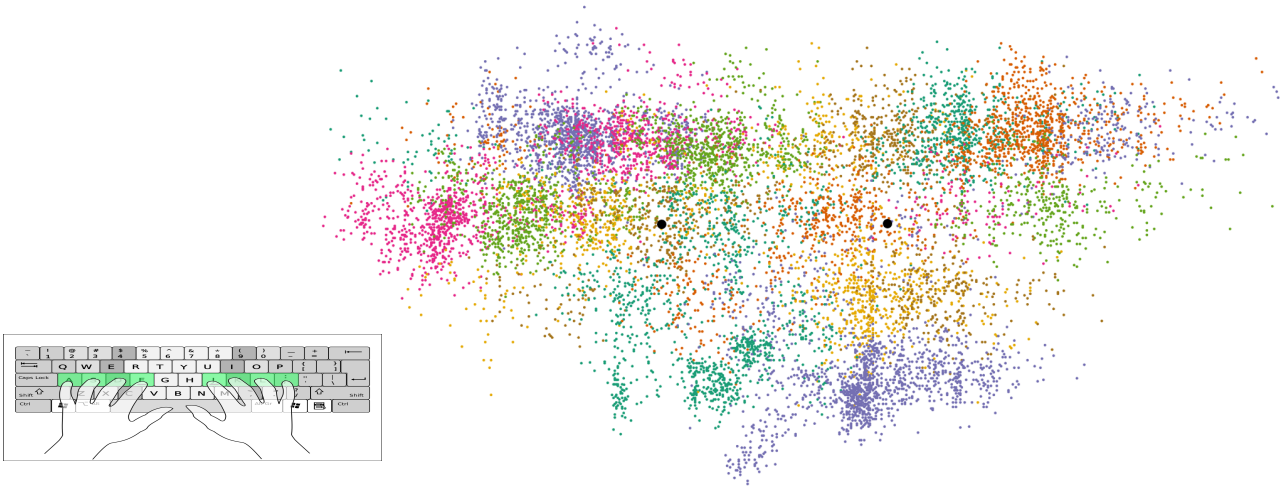


Fig. 4. Collective touch points merged across all participants, not scaled. Different colors indicate touch points from different keys. The two black dots show the centroids of key "F" and "J".

As expected, users' touch points for each key roughly followed a Gaussian distribution. We calculated SD_x and SD_y as the standard deviation of touch points for each character key in x and y dimensions respectively. On average, the collective SD_x and SD_y were 35.8mm (SD = 2.9) and 19.1mm (SD = 2.1) respectively; mean SD_x and SD_y for individual users were 15.1mm (SD = 2.4, range: 10.9~36.8) and 10.4mm (SD = 1.1, range: 8.5~19.4) respectively. Participants always yielded touch spreads that are greater in width than in height, and the individual spreads were significantly smaller than collective spreads. This implies that users could type fairly precisely, but the systematic offset varied across participants. Participants also had different levels of standard deviations. All the above results pointed to the necessity of a user-specific keyboard model.

4.5.3 Fitting Keyboard Model. We fitted the observed touch points according to Equation 6 to obtain the keyboard model for individual participants. We fitted the keyboard model for letter keys only. For the space key, we used a naïve Bayesian classifier.

To account for the variability of the absolute keyboard location on the surface for individual participants, we defined the origin point (0, 0) as the midpoint between the observed "F" and "J" key centroids for each participant. The X-axis directed to the right, while the Y-axis directed downward. We then used the centroids of "F" and "J" keys to represent the location of the left hand keyboard and the right hand keyboard, respectively. Therefore, a positive X-axis offset means the keyboard lies to the right of the origin point, and a positive Y-axis offset means that the keyboard lies below the origin point.

We then fitted the touch points to a standard keyboard to obtain the keyboard model parameters: keyboard size, keyboard location and input noise (measured by touch point deviation). The fitting was done for the left hand keyboard and the right hand keyboard separately. Because we used a standard keyboard for fitting, the fitted size was relative to that of a standard key. We thus called it **scale ratio** in the rest of this paper. Thus, a scale ratio of 1 indicated that the underlying model was as the same size as a standard keyboard. The bigger a scale ratio was, the bigger the underlying keyboard was.

Fitting Keyboard Size

Table 1 shows the average scale ratio for each hand in both X-axis and Y-axis. The average scale ratio among participants were around 1.20 for both hands and axes. This result suggested that the user keyboards tended to be always larger than the physical keyboard [9]. The biggest keyboard size was 1.43~1.88 times the size of the smallest keyboard. This showed individual participants had very different keyboard sizes.

Table 1. Fitted keyboard size (scale ratio) of each hand in X- and Y-axis with standard deviations and ranges.

		Average	SD	Range
X Axis	Left Hand	1.23	0.12	0.98~1.40
	Right Hand	1.20	0.15	0.97~1.43
Y Axis	Left Hand	1.19	0.17	0.94~1.54
	Right Hand	1.24	0.21	0.88~1.59

To examine the change of the keyboard size over time, we fitted keyboard models for each individual sentence. Figure 5 shows the scale ratio in X-axis and Y-axis for both hands. Linear fitting results showed that the scale ratio tended to increase over time in all four conditions. This implies that the underlying keyboard tended to become larger during typing, which highlights the necessity of a dynamic adapting algorithm.

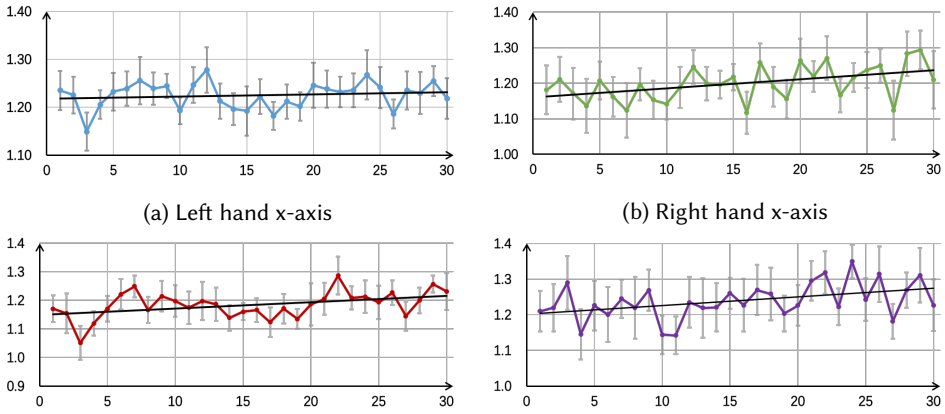


Fig. 5. Scale ratio plotted against sentence index. X-axis is the index of sentence; Y-axis is the corresponding avearge scale ratio for all users. The error bars show one standard error.

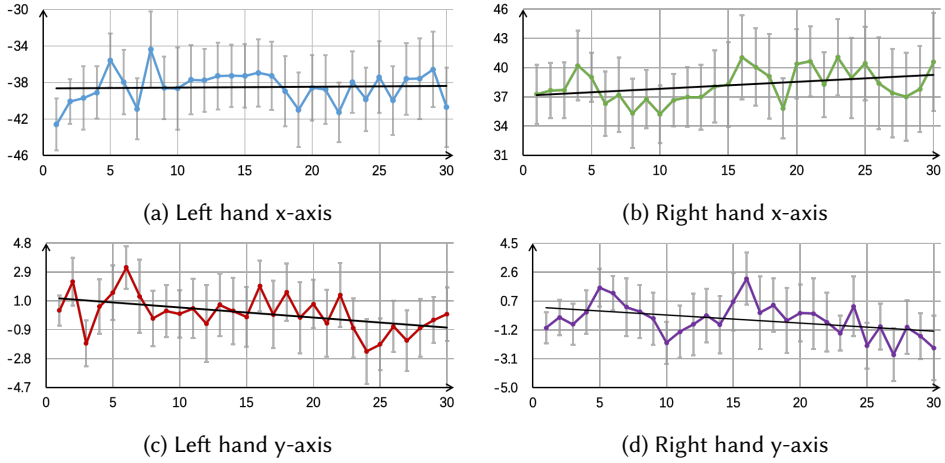
Fitting Keyboard Location

Table 2 shows the average offset for each hand and axis. In X-axis, the left hand keyboard and the right hand keyboard lied relatively symmetric to the origin point (the middle point). The average distance between "F" and "J" keys was 76.6mm (equal to the size of four keys on a standard keyboard). This suggested that the users were comfortable typing with more distance between their hands, perhaps to avoid hand collision. On the other hand, the amplitude of Y-axis offset was rather small (<1mm), implying that the touch points followed very well with the underlying keyboard model in Y-axis.

Table 2. Fitted keyboard location (mm) of each hand in X- and Y-axis with standard deviations and ranges.

		Average	SD	Range
X Axis	Left Hand	-38.2	13.6	-59.8~-12.4
	Right Hand	38.4	13.1	18.9~66.2
Y Axis	Left Hand	0.18	6.90	-18.4~8.7
	Right Hand	-0.34	5.89	-16.1~10.6

In addition, we investigated the change of the keyboard location over time, which indicated the effect of hand drifting. As shown in Figure 6, the left hand keyboard and right hand keyboard revealed similar trends. The X-axis offset tended to become larger over time, while the Y-axis offset tended to be smaller. This implied that both hands tended to move to the upper right during typing, which was consistent with [16].



(a) Left hand x-axis

(b) Right hand x-axis

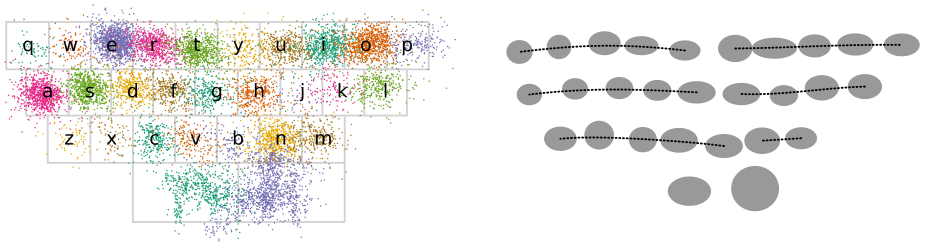
(c) Left hand y-axis

(d) Right hand y-axis

Fig. 6. Offset plotted against sentence index. X-axis is the index of sentence; Y-axis is the corresponding average offset (mm) of all users. The error bars show one standard error.

4.5.4 Keyboard Shape. We then scaled each participant's touch points to a standard size (Scale ratio = 1) according to their fitted keyboard model. Figure 7a showed the collective touch points after scaling, and Figure 7b shows the mean and standard deviation of the scaled touch spreads.

The emerged keyboard layout roughly followed a QWERTY layout. We also found that the emerged keyboard was slightly arched, especially the first row. We further fitted keyboard models using the centroid of the 26 keys.



(a) (b)

Fig. 7. (a) Collective touch points merged across all participants, scaled over a 1:1 keyboard. (b) Shape of keyboard, shows mean and one standard deviation along each dimension. Bézier curves fitted to the center of the spreads are shown in dotted lines.

The R^2 value were greater than 0.98 for both hands and axes, indicating a very good fit between the standard keyboard and the user keyboard emerged from observed touch points. This proved that users could touch type even without a visual keyboard, and justified validity of assuming the shape of the user keyboard as rectangle (See Assumption 2 in Section 3.2).

5 USER INPUT PREDICTION ALGORITHM

In this section, we describe the prediction algorithm based on users input to support high-performance eyes-free ten-finger typing on flat surfaces. Based on the results in experiment 1, we modified the MAP method of Goodman [11] to derive a hybrid algorithm. Different performances of different algorithms were compared through simulations. We also developed a dynamic adaptive algorithm to deal with the variation during typing.

5.1 Classical Bayesian Algorithm

The theory of input correction was first proposed by Goodman [11]. Given a sequence of input points I, the task is to select a word W in a predefined dictionary that is the most likely one. Goodman's model used a Maximum-a-Posteriori (MAP) method to calculate the likelihood of each word W according to I:

$$P(W|I) \propto P(I|W) \cdot P(W) \tag{8}$$

where $P(W)$ is a language model, which describes the frequency of the word, and $P(I|W)$ is a touch model, which quantifies the touch pattern of users. When calculating $P(I|W)$, traditional techniques usually consider individual touch points independently (e.g. [11]):

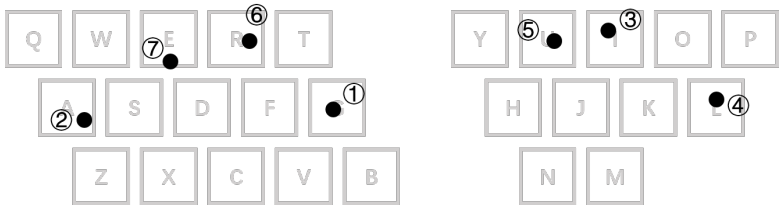
$$P(I|W) = \prod_{i=1}^{n} P(I_i|W_i) \tag{9}$$

where n is the length of the input sequence. I_i and W_i indicate the i-th touch point and character respectively. The calculation of Equation 9 relies on a key-press model for each individual keys. However, in the scenario of ten-finger eyes-free typing, this kind of model has three drawbacks. First, this model assumes an initial target location for each key, which is impossible to achieve when users can start to type anywhere on the surface.

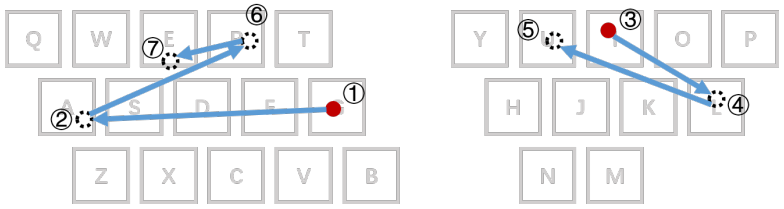
Second, the performance of this model is highly affected by hand drifting. In the previous experiment, we have showed that even for the same user, the location and size of the underlying keyboard model changed over time during eyes-free typing (see Figure 5 and Figure 6). Therefore, it is impossible to use a fixed position for each key.

Third, it treats each touch point independently, which did not fully leverage the information in the keyboard model. In Figure 4, we showed that users' touch points followed well with a QWERTY keyboard. Therefore, considering more touch points in calculation should be beneficial.

5.2 A Markov-Bayesian Algorithm



(a) A classical Bayesian algorithm (absolute). Absolute positions of touch points were used for decoding.



(b) A Markov-Bayesian algorithm (relative). Vectors between touch points within each hand were used. The first touch points (red points) were calculated by an absolute model.

Fig. 8. Illustration of information used by different algorithms when typing word "failure". The circled numbers indicate the sequential order of touch points performed by a user.

To address the problems mentioned in the last section, we modified the traditional method by considering keyboard model under two hands independently and augmented the method by treating the typing procedure as a Markov progress. That is, a left-hand keyboard and a right-hand keyboard model were established respectively, and the location of a touch point is affected by the location of the previous touch point in each half-keyboard. The differences of the classical and our new Markov-Bayesian algorithms are illustrated in Figure 8.

We split the keyboard into two parts by some middle column keys (i.e. T, Y, G, H, B, N), and these middle keys belonged to both left- and right-hand keyboards. This meant that users were not required to follow a standard touch typing finger position, where they could type these keys by either hand. Then in the preprocessing stage, we encoded each word in the corpus by the sequence of left/right hand used for each character. As a result, one word could be encoded by several different formats, and they are treated as different words in the calculation later. Then, for a sequence $W = W_1W_2 \cdots W_n$ with the corresponding "hand code", we could split the sequence into two parts: a left hand sequence $W_L = W_{L1}W_{L2} \cdots W_{La}$ (e.g. "fare" in Figure 8b) and a right hand sequence $W_R = W_{R1}W_{R2} \cdots W_{Rb}$ (e.g. "ilu" in Figure 8b), where $a + b = n$. Then we could calculate

$$P(I|W) = P(I_L|W_L) \cdot P(I_R|W_R) \tag{10}$$

where I_L and I_R were the corresponding subsequences in I as W_L and W_R in W.

For each probability in the right side of Equation 10, take left hand sequence as an example, we could calculate

$$P(I_L|W_L) = P(I_{L1}|W_{L1}) \cdot \prod_{i=2}^{a-1} P(I_{Li}, I_{L,i+1}|W_{Li}, W_{L,i+1}) \tag{11}$$

For the first tap, we calculate the probability using the absolute keyboard model. We fit an absolute keyboard model for both hands and both directions respectively. $P(I_{L1}|W_{L1})$ was calculated using a Bivariate Gaussian distribution for each key. The center of each distribution was the key center of the keyboard model, and the standard deviation in each dimension was the residual of the corresponding absolute model.

For the following taps, we calculated the probability using the relative keyboard model. The relative vectors between two consecutive touch points were used to calculate the probability (i.e. $P(I_{Li}, I_{L,i+1}|W_{Li}, W_{L,i+1})$, see illustration in Figure 8b). When calculating $P(I_{Li}, I_{L,i+1}|W_{Li}, W_{L,i+1})$, we calculate dx and dy as the deviation of two consecutive points in x and y dimensions respectively. We then calculated the probability using a bivariate Gaussian distribution according to the relative keyboard model and these two deviations. The mean of each distribution was the deviation between the key centers of the transition. The standard deviation in each dimension was the residual of the corresponding relative model.

By incorporating absolute touch positions and the relative movement pattern, TOAST can not only guess the initial position of the keyboard model, but also follow the change of the underlying keyboard model. Along with the help of language model, we can get satisfying results, as we will verify in the following parts.

5.3 Comparison of Different Algorithms

The Markov-Bayesian algorithm assumed that leveraging the relative movement pattern was beneficial for the performance. In this section, we performed a series of simulation on the data we collected in the previous experiment. The goal is to determine the best strategy for our algorithm. The available selections are:

Absolute vs. Relative When calculating $P(I|W)$, a traditional way is to use an absolute keyboard model [8, 10, 11]. This model describes the pattern of touch point distribution for each individual keys. In calculation, each input point is treated independently.
In the contrary, the Markov-Bayesian model (relative model) leveraged the relative movement pattern in calculation. It describes the pattern when users move from key A to key B, which is not so affected by hand drifting. Therefore we expect that the Markov-Bayesian model could outperform a simple absolute keyboard model.

General vs. Per User Personalization is always a top topic for soft keyboards (e.g. [8, 13, 18]). In Table 2, Table 1 and Figure 4, we have showed that the touch pattern varied significantly among individual users. Also, the underlying keyboard model of different users varied in size and offset. Therefore, we expect that per-user model would yield better performance than a general keyboard model.

We performed a simulation on the collective data in the previous experiment. We tested the performance of different keyboard models using 5-fold cross validation. We tested the two factors mentioned above, resulting in four kinds of keyboard models: general × absolute, general × relative, per-user × absolute, and per-user × relative. We ran the algorithms on the input stream of each individual users, and analyzed the word-level accuracy in top1, top3 and top5 candidates.

Figure 9 shows the average word-level accuracy of each algorithm. As expected, per-user algorithms outperformed general algorithms significantly. The accuracy of the relative general model is significantly higher than that of the absolute general model, while the two per-user algorithms yielded very similar results. Noticeably, the two per-user algorithms both yielded a top1 accuracy of over 96%, and a top3 accuracy over 99%. This confirmed that the Markov-Bayesian model did explained the noise in the typing data. According to these results, per-user relative algorithm should be used in a real typing technique.

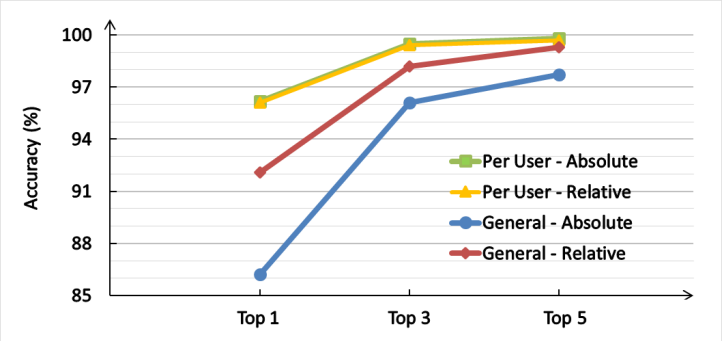


Fig. 9. Word-level accuracy of each algorithm, shows top1, top3 and top5 candidate.

5.4 Adaptive Strategy

In Figure 5 and Figure 6, we showed that even for the same participant, the parameter of the underlying changed with time. In order to address this problem, we also employed a dynamic adaptive strategy for TOAST.

When a user started to type with TOAST, an initial keyboard model would be used. During typing, the dynamic adaptive strategy used a time window to collect the typing data that users generate for the latest typed words, and update the keyboard models according to the data. We used the selected typing words as the user's approximate intention words for we would never know the ground truth input words of the user. Each time the user confirmed the selection of a word, TOAST would fit a new keyboard model using the data of the latest N words, and use it to replace the older models. The adaptive method was a least squares fitting shown by Equation 6, as described in Section 3.3.

Of course, the selection of N is crucial for the performance of the algorithm. If N is too large, the algorithm may adapt too slow. However, if N is too small, the algorithm may be too sensitive and may over-fit data noise. In order to find an appropriate value of N, we performed another simulation using different N on the data in the previous experiment. We took the input stream of each user as the input, and calculated the average word-level accuracy of top 1,3,5 candidates across all participants. We chose 5 levels of N for the simulation according to our data size: 5, 10, 15, 20 and 25.

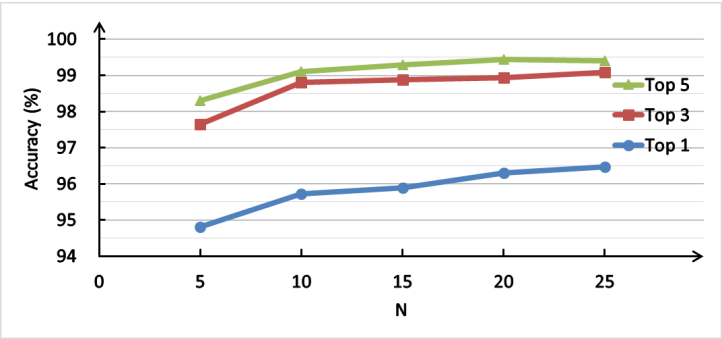


Fig. 10. Word-level accuracy in different N. Shows top1, top3 and top5 candidates.

Figure 10 shows the average word-level accuracy for different N. The accuracy in 10 was significantly higher than that in 5, but bigger N would not further improve the performance significantly. Therefore, 10 words should be adopted as the window width of the dynamic adaptive strategy.

6 EVALUATION

In this section, we evaluated the performance of TOAST through a second experiment. First, we designed interaction and algorithm using the results in the first experiment, and implemented a prototype of TOAST. Then we asked participants to perform text entry tasks with TOAST. We were interested in the performance that TOAST could achieve in real use scenarios, as well as the user preferences.

6.1 The Design of TOAST

We designed the interaction of TOAST as following steps:

(1) A user put both his/her hands on the surface, and put all his/her eight fingers on the home row of the imaginary keyboard (i.e. “A”, “S”, “D”, “F” keys for left hand; “J”, “K”, “L”, “;” keys for right hand). The user then tap both thumbs simultaneously to trigger calibration, which would place the initial keyboard model under his/her left and right hands respectively.
(2) The user then perform ten-finger eyes-free typing with TOAST. There is no visual keyboard. Instead, an underlying keyboard model would be used for each hand respectively, which would recognize the target word.
(3) Each time a tap event occurred, the system would show five candidate words with the highest likelihood. The user could tap left thumb to circularly select words from the candidate list, and tap right thumb to confirm the selection. The selected word will be appended to the input box after confirmation and a space would be appended after the selected word.
(4) During typing, the user could also swipe either thumb from right to left to delete the last input word.

We also implemented our algorithm based on the results in Section 5.1. That is, we adopted a per-user relative Markov -Bayesian algorithm, combined with an dynamic adaptive strategy with a window with of 10 words.

6.2 Participants

We recruited 16 participants from the campus (15 male, 1 female), with an average age of 21.8 (SD = 1.7). None of them have participated in the first experiment. All of them regularly use a QWERTY keyboard. We evaluate the expertise of their typing ability on a physical keyboard using TextTest [34]. They typed an average of 68.4 WPM (SD = 18.2) with an uncorrected error rate [29] of 0.3% (SD = 0.3%). Each participant was paid $30.

6.3 Experiment Design

The experiment apparatus was the same as in the first experiment, except that users used a prototype of TOAST to complete real text entry tasks this time. We implemented the input prediction, dynamic adaptation algorithms and interaction designs mentioned above. We also put a space key below the keyboard of each hand (see Figure 11). The width of the gap between the keyboard and the space keys were empirically set to 0.5 key width. During experiment, the task texts were shown right above the input box (not shown in Figure 11). After finishing each sentence, the experimenter would continue to the next sentence by hitting the space key on a physical keyboard.

We used the top-30,000 word in the ANC data [1] as our language model. Users completed four blocks of text entry tasks in total. In each block, users entered 10 phrases sampled randomly from the MacKenzie and Soukoreff corpus [21]. We ensured that each phrase comprised only words within the vocabulary. In total, 16 participants × 4 blocks × 10 phrases/block = 640 phrases were entered.

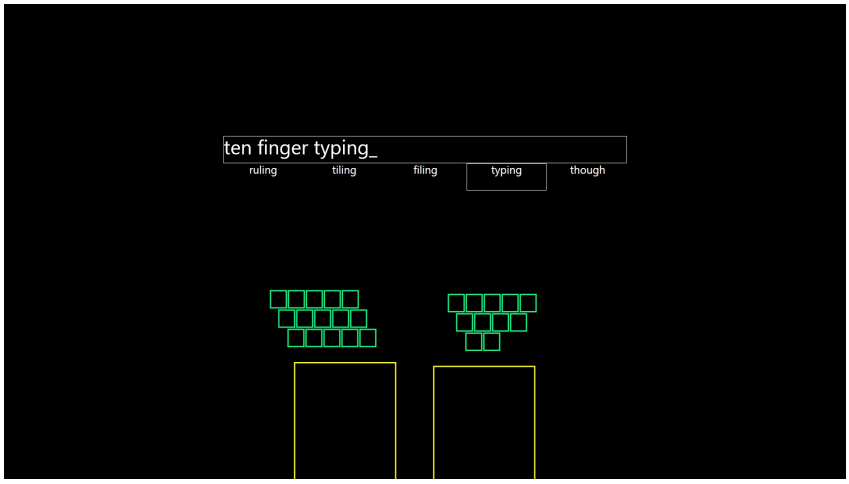


Fig. 11. TOAST interface. The underlying keyboard model(green) and the space region(yellow) were not visible to the users. The candidate list was shown below the text area. During experiment, there would be task texts shown above the input box.

We did not choose a baseline technique for comparison in our second experiment for the following reasons: first, we defined the scenario of TOAST as a ten-finger eyes-free typing on flat surface, which were not mentioned for any existing techniques to our knowledge. Second, in the practical usage, we found absolute model, relative model as well as the adaptive mechanism were all essential for TOAST to work. Users found it difficult to type even one word without any of the components. Therefore, it was not suitable to compare our technique with a simple one.

6.4 Procedure

Participants first completed a questionnaire that collected their demographic information. They were then allowed 5 minutes to familiarize themselves with the interaction of TOAST. Next, they were asked to complete 4 blocks of text entry tasks. During the experiment, they were told to "type as fast and as accurately as possible" and were allowed to delete or retype the word if an error occurred. A two-minute break was enforced between blocks. Finally, questionnaire and interviews were carried out to gather subjective feedback.

6.5 Results

6.5.1 Speed. The average speed was 43.1 WPM (SD=9.5 WPM) across all blocks, which was 2/3 of the speed on physical keyboards. Compared with existing techniques (e.g. 30WPM in [8], 38.68 WPM in [14]), our technique provideed 11% to 44% higher speed even under the eyes-free condition, which indicated more challenge for typing (mentioned in Section 1). The fastest participant even reached 66.1 WPM in average, with a peak speed at 83 WPM. The speed was correlated to the input speed on a physical keyboard (R^2=0.63). This indicated that being expert for typing on physical keyboard could improve the performance of typing by TOAST.

Figure 12 shows the average speed for each block. Speeds over four blocks were 41.4 (SD=9.5), 43.6 (SD=11.0), 42.6 (SD=9.8) and 44.6 (SD=8.8) WPM respectively. RM-ANOVA showed a significant effect of block on speed ($F_{3,45} = 3.20, p < .05$), *post hoc* tests showed that speeds in block 2 and 4 were significant higher than that in block 1 ($F_{1,15} = 4.98,\ p < .05$ and $F_{1,15} = 9.45,\ p < .05$), suggesting that speed could improve after some practice. The valid practice time was less than ten minutes.

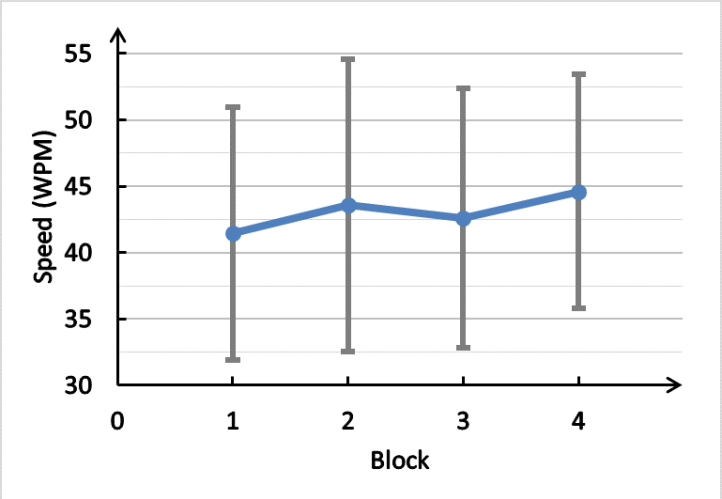


Fig. 12. Text entry speed in each block.

6.5.2 Error Rate. We computed the character-level error rate using CER, which can be interpreted as the minimum edit distance between the task string and the final transcribed string, divided by the length of the task phrase. Users tended to fix most, if not all, of their errors during typing, leaving few in the final transcribed string. The average CER across blocks was 0.6% (SD = 0.4%). The CER in each blocks were 0.6% (SD=0.8%), 0.9% (SD=0.9%), 0.4% (SD=0.6%) and 0.6% (SD=0.6%) respectively. RM-ANOVA found no significant effect of block on CER ($F_{3,45} = 1.39$, $p = .26$).

6.5.3 Interaction Statistics. In this section, we look more into the word-affecting interactions that users perform during the experiment. Table 3 shows the number and ratio of all word-affecting interactions.

Table 3. Number(ratio) of all word-affecting actions. Topk(k=1,2,3,4,5) indicates the action that a user selected the k-th candidate in the candidate list. Correctness indicates whether the selected candidate matches the target word. All the ratios sum up to 100%.

Correctness \ Action	Top1	Top2	Top3	Top4	Top5	Delete
Correct	2788(78.6%)	63(1.8%)	19(0.5%)	3(0.1%)	4(0.1%)	324(9.1%)
Incorrect	321(9.1%)	15(0.4%)	2(0.1%)	4(0.1%)	2(0.1%)	

The most common action was Top1 (78.6%): selecting the first candidate and confirming the selection. This showed that the prediction accuracy of TOAST was very high. Noticeably, 9.1% of the selection was incorrect. We speculated that users were confident in the correction ability of TOAST, thus tended to select the first candidate in most cases. When the target word was not the first candidate, this would lead to a false selection and a following deletion (in most cases).

6.5.4 Qualitative Results of User Preference. In this section we summarize the subjective feedback from user interviews.

Though our participants were not used to eyes-free ten-finger typing on a flat surface at first, they could learn to type quickly.

"At first, it was really hard for me to type without a visual keyboard, but the keyboard dynamically followed the movement of my hands, which makes typing very easy." (P2)

Though there was no visual feedback, participants could type at a high speed with acceptable accuracy.

"It is amazing that I can type so fast on a touchscreen. And I can type words correctly even when I missed some keys." (P9)

The design of space key was natural for users, but the reliability of this action is very important.

"I like to select and confirm words with my thumbs. That is very natural. But it bothers me when the system confused space key and character keys."(P13)

As we perform word-level corrections, the target word would not appear until the final character have been entered. This might have caused extra mental efforts for some users.

"I have to type in every character to see whether the input is correct. I wish word prediction could be supported." (P6)

7 LIMITATIONS AND FUTURE WORK

Though we have conducted understandings and explored algorithms for eyes-free ten-finger typing algorithm, as well as evaluated the feasibility of our algorithm, there still exists some limitations and future work.

In the description of keyboard model, we only considered the change of keyboard size and location, and assumed keyboard is rectangle and horizontally placed (Assumptions 1 and 2 in Section 3.2). To achieve more natural and free experience for typing on flat surfaces, we can extend the algorithm to account for more variables such as shape and orientation.

Our technique is designed for those who can already perform eyes-free typing on the physical keyboard. The feasibility of novice typists using our technique by some aiding (e.g. rendering the keyboard layout on the information display) still needs to be explored.

In the evaluation experiment, we recruited only 1 female participant, which may affect the results. However, the gender distribution was more balanced in the first experiment, and we have verified the results through several simulations using their data before performing the evaluation. Our design and implementation of algorithm were based on the simulation results, for which we believed the influence of unbalanced participants should be weak.

Currently, our algorithm can only predict words that have equal length to the input sequence, and we can only interact with the system by some simple gestures. It is important to support unaligned inputs with auto-correction and auto-completion. This can be served as another way to deal with the input noise caused by both human and the device, and thus allow users to type even faster. Non-alphabetic characters (e.g. numbers, punctuations) and function keys (e.g. shift, return keys) are also important for text input, and may be added in the future (e.g. by means of some multi-touch gestures [7]). More natural interactions (e.g. allowing users to rest on the surface to reduce fatigue) can be added in the future for a better experience.

We used 2D touch data collected from a Microsoft Surface to develop our algorithm in this paper. We described that this environment could be extended to any surface with certain sensing technique that could report touch points. We can further experiment our technique with other touch sensing techniques, for example, a computer vision-based touch detection system (e.g. a Kinect or Leap Motion), or capacitive touch screens. We can also experiment our technique with other surfaces, like on clothing, pillows, walls, even thighs. These new sensing techniques and new surfaces can provide touch point data of different quality, with more context information or different ubiquitous experience, which may lead to different results.

8 DISCUSSION AND CONCLUSION

In this paper, we present a novel algorithm, TOAST, for enabling ten-finger typing on flat surfaces, with limited tactile feedback, with no visual feedback and with no constraint on the input surface. The TOAST algorithm reflects our deep understanding about users' eyes-free typing behavior via a keyboard model, and advances the-state-of-the-art text entry techniques to deal with different source of variations when typing on flat surface. Two improvements are made: the first is to predict users' input based on the relative offsets of successive touch points, to deal with the uncertainty of the keyboard location. The second is to dynamically adapt the underlined keyboard model in that updates are made for the entire keyboard rather than individual keys. Therefore, the adaptation can be done very fast (with a few touch points) and is more insensitive to input noise.

Our first experiment investigates users' typing behavior by fitting observed touch data to a keyboard model. Compared to previous researches, the obtained result highlights the difference between users and change of keyboard parameters during different typing sessions within users. Meanwhile, our simulation study provides theoretical insights about the role of different design factors for the accuracy of prediction algorithm. The result suggests the necessity of a user-specific algorithm, and the advantage of predicting based on vectors.

Evaluation results showed users could type at 45 WPM (67% of the typing speed on a physical keyboard) with limited learning. This result is significantly faster than all previous techniques (e.g. 30 WPM in [8], 38.68 WPM in [14]) concerning ten-finger typing on a flat surface. This result showed human can transfer their typing skill from a physical keyboard to a flat surface with limited tactile feedback, and it is feasible to interpret users' typing intention with sophisticated algorithm.

ACKNOWLEDGMENTS

This work is supported by the Natural Science Foundation of China under Grant No. 61521002, 61672314 and No. 61572276, National Social Science Fund of China under Grant No. 15CG147, Tsinghua University Research Funding No. 20151080408, and also by Beijing Key Lab of Networked Multimedia.

REFERENCES

[1] 2011. The Open American National Corpus(OANC). (2011). http://www.anc.org
[2] 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*. ACM, 251–260.
[3] 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*. ACM, 57–66.
[4] Daewoong Choi, Hyeonjoong Cho, and Joono Cheong. 2015. Improving Virtual Keyboards When All Finger Positions Are Known. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM, 529–538.
[5] James Clawson, Kent Lyons, Alex Rudnick, Robert A Iannucci Jr, and Thad Starner. 2008. Automatic whiteout++: correcting mini-QWERTY typing errors using keypress timing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 573–582.
[6] Francine Evans, Steven Skiena, and Amitabh Varshney. 1999. VType: Entering text in a virtual world. *submitted to International Journal of Human-Computer Studies* (1999).
[7] Leah Findlater, Ben Lee, and Jacob Wobbrock. 2012. Beyond QWERTY: augmenting touch screen keyboards with multi-touch gestures for non-alphanumeric input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2679–2682.
[8] Leah Findlater and Jacob Wobbrock. 2012. Personalized input: improving ten-finger touchscreen typing through automatic adaptation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 815–824.
[9] Leah Findlater, Jacob O Wobbrock, and Daniel Wigdor. 2011. Typing on flat glass: examining ten-finger expert typing patterns on touch surfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2453–2462.
[10] Kentaro Go and Yuki Endo. 2007. CATKey: customizable and adaptable touchscreen keyboard with bubble cursor-like visual feedback. *Human-Computer Interaction–INTERACT 2007* (2007), 493–496.
[11] 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*. ACM, 194–195.

[12] 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*. ACM, 3817–3821.

[13] Asela Gunawardana, Tim Paek, and Christopher Meek. 2010. Usability guided key-target resizing for soft keyboards. In *Proceedings of the 15th international conference on Intelligent user interfaces*. ACM, 111–118.

[14] Sunjun Kim, Jeongmin Son, Geehyuk Lee, Hwan Kim, and Woohun Lee. 2013. TapBoard: making a touch screen keyboard more touchable. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 553–562.

[15] Falko Kuester, Michelle Chen, Mark E Phair, and Carsten Mehring. 2005. Towards keyboard independent touch typing in VR. In *Proceedings of the ACM symposium on Virtual reality software and technology*. ACM, 86–95.

[16] Frank Chun Yat Li, Leah Findlater, and Khai N Truong. 2013. Effects of hand drift while typing on touchscreens. In *Proceedings of Graphics Interface 2013*. Canadian Information Processing Society, 95–98.

[17] Frank Chun Yat Li, Richard T Guy, Koji Yatani, and Khai N Truong. 2011. The 1line keyboard: a QWERTY layout in a single line. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 461–470.

[18] 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. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 2 (2017), 18.

[19] Kent Lyons, Daniel Plaisted, and Thad Starner. 2004. Expert chording text entry on the twiddler one-handed keyboard. In *Wearable Computers, 2004. ISWC 2004. Eighth International Symposium on*, Vol. 1. IEEE, 94–101.

[20] I Scott Mackenzie. 2002. A note on calculating text entry speed. *Unpublished work. Available online at http://www.yorku.ca/mack/RN-TextEntrySpeed.html* (2002).

[21] 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*. ACM, 754–755.

[22] I Scott MacKenzie, Shawn X Zhang, and R William Soukoreff. 1999. Text entry using soft keyboards. *Behaviour & information technology* 18, 4 (1999), 235–244.

[23] Taichi Murase, Atsunori Moteki, Genta Suzuki, Takahiro Nakai, Nobuyuki Hara, and Takahiro Matsuda. 2012. Gesture keyboard with a machine learning requiring only one camera. In *Proceedings of the 3rd Augmented Human International Conference*. ACM, 29.

[24] Daniel R Rashid and Noah A Smith. 2008. Relative keyboard input system. In *Proceedings of the 13th international conference on Intelligent user interfaces*. ACM, 397–400.

[25] Helena Roeber, John Bacus, and Carlo Tomasi. 2003. Typing in thin air: the canesta projection keyboard-a new method of interaction with electronic devices. In *CHI'03 extended abstracts on Human factors in computing systems*. ACM, 712–713.

[26] Robert Rosenberg and Mel Slater. 1999. The chording glove: a glove-based text input device. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 29, 2 (1999), 186–191.

[27] Christian Sax, Hannes Lau, and EM Lawrence. 2011. Liquid Keyboard: An Ergonomic, Adaptive QWERTY Keyboard for Touchscreens and Surfaces. In *The Fifth International Conference on Digital Society*. IARIA Conference.

[28] Oliver Schoenleben and Antti Oulasvirta. 2013. Sandwich keyboard: fast ten-finger typing on a mobile device with adaptive touch sensing on the back side. In *Proceedings of the 15th international conference on Human-computer interaction with mobile devices and services*. ACM, 175–178.

[29] R William Soukoreff and I Scott MacKenzie. 2004. Recent developments in text-entry error rate measurement. In *CHI'04 extended abstracts on Human factors in computing systems*. ACM, 1425–1428.

[30] Caleb Southern, James Clawson, Brian Frey, Gregory Abowd, and Mario Romero. 2012. An evaluation of BrailleTouch: mobile touchscreen text entry for the visually impaired. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*. ACM, 317–326.

[31] Keith Vertanen, Haythem Memmi, Justin Emge, Shyam Reyal, and Per Ola Kristensson. 2015. VelociTap: Investigating fast mobile text entry using sentence-based decoding of touchscreen keyboard input. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 659–668.

[32] Daryl Weir, Henning Pohl, Simon Rogers, Keith Vertanen, and Per Ola Kristensson. 2014. Uncertain text entry on mobile devices. In *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 2307–2316.

[33] Malte Weiss, Roger Jennings, Ramsin Khoshabeh, Jan Borchers, Julie Wagner, Yvonne Jansen, and James D Hollan. 2009. SLAP widgets: bridging the gap between virtual and physical controls on tabletops. In *CHI'09 Extended Abstracts on Human Factors in Computing Systems*. ACM, 3229–3234.

[34] Jacob O Wobbrock and Brad A Myers. 2006. Analyzing the input stream for character-level errors in unconstrained text entry evaluations. *ACM Transactions on Computer-Human Interaction (TOCHI)* 13, 4 (2006), 458–489.

[35] Xin Yi, Chun Yu, Weinan Shi, and Yuanchun Shi. 2017. Is It Too Small?: Investigating the Performances and Preferences of Users when Typing on Tiny QWERTY Keyboards. *International Journal of Human-Computer Studies* (2017).

[36] Xin Yi, Chun Yu, Mingrui Zhang, Sida Gao, Ke Sun, and Yuanchun Shi. 2015. Atk: Enabling ten-finger freehand typing in air based on 3d hand tracking data. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM, 539–548.

[37] Ying Yin, Tom Yu Ouyang, Kurt Partridge, and Shumin Zhai. 2013. Making touchscreen keyboards adaptive to keys, hand postures, and individuals: a hierarchical spatial backoff model approach. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2775–2784.

Received November 2017; revised January 2018; accepted January 2018