Bimanual Gesture Keyboard

Xiaojun Bi **Ciprian Chelba** **Tom Ouyang** **Kurt Partridge** **Shumin Zhai**
Google Inc.
Mountain View, CA, USA
{bxj, ciprianchelba, ouyang, kep, zhai}@google.com

ABSTRACT
Gesture keyboards represent an increasingly popular way to input text on mobile devices today. However, current gesture keyboards are exclusively unimanual. To take advantage of the capability of modern multi-touch screens, we created a novel bimanual gesture text entry system, extending the gesture keyboard paradigm from one finger to multiple fingers. To address the complexity of recognizing bimanual gesture, we designed and implemented two related interaction methods, *finger-release* and *space-required*, both based on a new multi-stroke gesture recognition algorithm. A formal experiment showed that bimanual gesture behaviors were easy to learn. They improved comfort and reduced the physical demand relative to unimanual gestures on tablets. The results indicated that these new gesture keyboards were valuable complements to unimanual gesture and regular typing keyboards.

Keywords: Text Entry, Touch Screen

ACM Classification Keywords
H.5.2 [Information Interfaces and Presentation]: Input Devices and Strategies

INTRODUCTION
Among numerous novel methods of text input, gesture keyboards, also referred as shape writing [26, 21], have become a rare exception in gaining large-scale adoption on mobile devices. First published in the user interface literature in the early 2000's [27, 13], gesture keyboards today are commercially available through many products such as ShapeWriter, SlideIT, Swype, T9 Trace, and TouchPal.

A gesture keyboard offers users a variety of advantages. It supports a gradual and seamless transition from visually guided tracing to recall-based gesturing, and also relaxes the requirement of precisely specifying words verbatim by allowing users to express the intention with approximate shape and location finger strokes [13]. Also, it is immune to one major problem plaguing regular touchscreen typing: the lack of tactile-feedback [7, 21].

One frequently raised question is that given the capability of modern multi-touch screens, it seems amiss that gesture keyboards are exclusively unimanual. Traditional typewriting is a two-handed activity. Furthermore, many mobile device users often hold the device with both hands and type with two thumbs (Figure 1). On many tablets the virtual keyboard can be split to two parts to support this mode of operation. Additionally, a split keyboard consumes only a small amount of screen space, which has been shown to boost user experience on portable touch screens. [15]. For example, the sizes of the split keyboards in Figures 1a and 1b are only 25% (landscape) and 47% (portrait) of the sizes of default virtual tablet keyboards. In recognition of these advantages, split keyboards are now supported on iPad, Windows 8 and Android tablets. However, despite the popularity of split keyboards, current gesture keyboards are incompatible with two-thumb use. We aim to overcome this limitation by enabling bimanual operation of gesture keyboards.

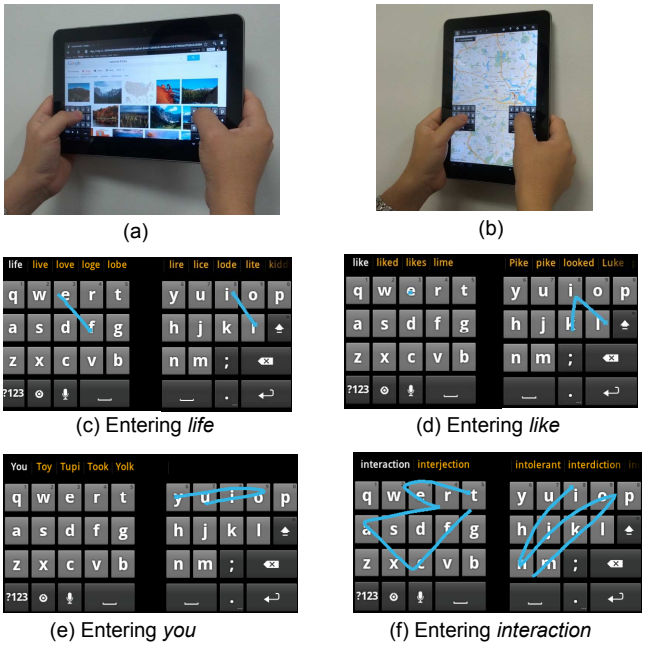


Figure 1. (a) and (b): two-hand holding postures for tablets with split keyboards. (c), (d), (e), and (f): entering words on split keyboards with bimanual gestures. Note that the actual gap between the two sides of a split keyboard is greater than illustrated. Blue strokes represent finger strokes.

Other than supporting two-handed holding and typing, and freeing screen real estate, another benefit of enabling bimanual operation lies in movement efficiency. Instead of frequently moving the same finger from one side of the

UIST '12, October 7–10, 2012, Cambridge, Massachusetts, USA.

keyboard to another, each thumb only moves in its vicinity on one side of the keyboard, hence eliminating the long movements common on unimanual gesture keyboards. For example to enter the word *life*, a unimanual gesture approximates the trace *l-i-f-e* on the keyboard. On a standard Qwerty keyboard the trace from *i* to *f* moving from the right side to the left side of the keyboard is the longest segment of the *l-i-f-e* gesture. With a bimanual gesture keyboard, the word *life* can be gestured with *l-i* movement by the right hand and *f-e* by the left hand, eliminating the *i-f* segment (Figure 1). In another example, the bimanual gesture for the word *like* can be drawn as *l-i-k* by the right hand and a tap (a zero length gesture) on *e* by the left hand, saving the long distance travel from *k* to *e* on Qwerty (Figure 1).

It is important and in fact necessary to design bimanual gesture keyboards to be fully compatible with existing modes of keyboard operation. First, bimanual gestures algorithm should be compatible with unimanual gestures. Any additional flexibility or benefits that bimanual gestures may bring should be at little or no cost of unimanual gestures. Second, tapping is still the preferred mode of touch screen keyboard operation for many users today (if not the majority). Gesture keyboards should not be viewed as competition or as a replacement for tapping. They should be compatible with the tapping mode of keyboard as well.

The contribution of this paper is two-fold. First, we designed and implemented a bimanual gesture text entry system, based on the *multi-stroke gesture recognition* algorithm as well as two interaction methods (i.e., *finger-release* and *space-required*). This system enabled a novel form of bimanual gesture input that was not previously possible, and extended the gesture keyboard from one finger to multiple fingers. Second, we conducted a formal experiment to evaluate the strengths and weaknesses of bimanual gesture input. The results showed that the bimanual behavior was easy to learn. It raised the comfort level and reduced the physical demand over the unimanual counterpart. 42% (15 out of 36) participants chose one of the bimanual gesture keyboards as the most preferred input method. The weakness was that it had slightly inferior speed efficiency over unimanual gesture keyboard, due to the attention and action switches between the two fingers.

In the rest of this paper we first briefly review related work. We then focus on the main bimanual gesture algorithms and the systems we have developed, outlining the pros and cons of each design. We then report and analyze a formal lab experiment evaluating bimanual gesture keyboards on tablets.

RELATED WORK

To our knowledge no bimanual gesture keyboard research has been reported in the literature to date.

In the body of literature on unimanual gesture keyboards, Zhai and Kristensson published the first prototype, experiment and a set of basic principles of gesture keyboards [27]. The first complete and large vocabulary gesture keyboard was published in [13]. Kristensson and Zhai [12] extended the gesture keyboard concepts to gesture keyboard commands whose strokes are defined by the command name but prefixed on a command key. Many of the key concepts of gesture keyboard were summarized in "Introduction to Shape Writing" [26]. The idea of treating typing as a discrete form of "shape writing" was reported in the work of Elastic Stylus Keyboard (ESK) [14] . Zhai et al analyzed user feedback of ShapeWriter on the iPhone [28]. Rick [21] developed a predictive modele of gesture keyboard and used such a model to estimate keyboard layout's impact on gesture keyboard performance.

A large body of research on two-handed interaction interfaces, beginning with Buxton and Myers [7], can be found in the HCI literature. A focus of this body of work is on the task assignment between two hands [e.g. 4]. The best known model of bimanual action is Guiard's kinematic chain theory [9] explaining the asymmetric relationship between the dominant and the non-dominant hand's role bimanual operation. In this theory the non-dominant hand sets the coarse level frame of reference (e.g. holding a pad) and the dominant hand makes fine control within that frame of reference (e.g. writing with a pen on the pad). In human-computer interaction as well as other real world tasks bimanual operation can also be symmetrical (typing, steering a bicycle). The literature suggests that a number of tasks such as 2D or 3D navigation [3] can be performed effectively with a symmetric assignment of roles to the hand. Factors affecting the performance of symmetric bimanual interaction were also studied [2]. In the field of text entry, symmetric bimanual typing is common on both physical and virtual keyboards. In this work, we extend the bimanual interaction to gesture keyboards.

There is also a long history of research on gesture and sketch based interfaces in the HCI community. These gesture symbols can be drawn using either a single stroke [22, 16] or multiple strokes [1, 19], and have been applied to problems ranging from interfaces for blind users [11] to search on mobile devices [17, 20]. In contrast to our work, these approaches focus on unimanual input where strokes are often drawn sequentially. Furthermore, they represent an inherently different type of interaction where gestures are direct visual recreations of the symbols in the domain, rather than as paths that are defined by virtual keys on a soft keyboard. Finally, strokes are recognized at the word-level on gesture keyboards but mostly at the character-level in other applications.

BIMANUAL GESTURE KEYBOARDS

A bimanual gesture keyboard works similarly to a unimanual keyboard: a word is entered by tracing through all the letters, from the first to last in order. Unlike the unimanual gesture keyboard, on which a word is entered by one continuous stroke, a user can draw multiple strokes for a given word on a bimanual keyboard using both thumbs (Figure 1). In this paper, we define a *stroke* as a continuous trajectory drawn by a finger on the touch screen, with one

touch down, multiple *touch move,* and one *touch up* events. A *gesture* refers to a collection of *strokes* for a given word. A *unimanual gesture* contains only one *stroke,* while a *bimanual gesture* can contain multiple *strokes* (Figure 1).

As the number of strokes in a *bimanual gesture* is uncertain and previous unimanual gesture algorithms only process single-stroke *gestures*, there are two new challenges to recognizing bimanual gestures: 1) how to *cluster* the *strokes* that belong to the same word into a bimanual gesture and 2) how to *recognize* the target word that the bimanual gesture represents.

Clustering Strokes

On bimanual gesture keyboards, a word can be represented by one or multiple *strokes*, depending on the locations of letters in a word. For example, the word *you* is usually drawn by one *stroke*, because letters *y*, *u* and *o* are on the same side of the keyboard and close to each other, while *"life"* is usually represented by two *strokes*, *l-i* and *f-e*, because *l, i,* and *f, e* are located at different sides and drawn by two thumbs separately. *Interaction* is usually represented by three strokes: *i-n (right), t-e-r-a-c-t (left),* and *i-o-n (right)* (Figure 1).

To recognize words from bimanual gestures, the first step is cluster *strokes* into *gestures*. To be compatible with unimanual gesture and two-thumb typing, and transfer users' experience of using these two methods, we propose two approaches to solve this problem: *Finger-Release* and *Space-Required*.

Finger-Release. Similar to unimanual gestures, which signal the end of a word by the lifting up the finger, this method uses *finger-release* to cluster gestures: lifting both fingers off the screen indicates the end of a word. In other words, the user has to keep at least one finger on the screen while composing the word.

This method is fully compatible with unimanual gesture keyboard. It also allows users to skip inputting the space key, which is a big advantage of gesture keyboards in general.

The disadvantage is that it may require extra mental and physical efforts for users to keep at least one finger on the screen before finishing the word. It may be difficult to get used to such a behavior. If the user accidentally releases both fingers when switching one finger to the other, it would falsely signal the end of a word.

Space-Required. Another approach is to use a space key press to signal the end of a word, the standard method for regular tap-based keyboards.

This method offers users more flexibility than *Finger-Release*, by removing the requirement of keeping at least one finger down. It is also fully compatible with regular typing. Our algorithm even supports mixing regular typing and gesturing for a given word, since both gesturing and typing use the space key press to signal the end of a word. This method can also be viewed as enhanced typing: a user can choose to tap, gesture, or mix tapping and gesturing for a single word.

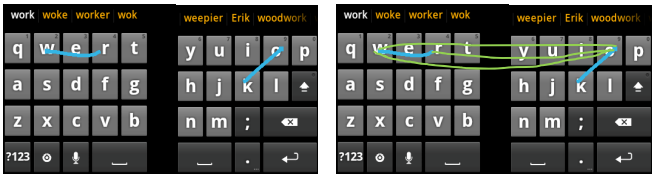
The disadvantage is that it is somewhat incompatible with unimanual gesture by requiring a space key after finishing gestures. If both unimanual and bimanual gestures are implemented on the same keyboard, it requires pressing a space key for unimanual gestures too.

In short, *finger-release* and *space-required* have their own pros and cons as stroke clustering methods. It is unknown which one fits bimanual gesture keyboards better. We implemented both of them and conducted a formal lab study to understand the strengths and weaknesses of each.

Multi-Stroke Gesture Recognition

A *bimanual gesture* usually consists of *multiple strokes.* To our knowledge, no literature shows how to recognize the word from multiple strokes in the gesture keyboard paradigm.

A straightforward approach is to connect all touch points together to form a unimanual gesture according to the time stamps, and feed it into a unimanual gesture recognition engine [13]. This approach works if all the letters of a word happen to be on the same side of the screen. (e.g., the word *you*, all three letters are on the right side). However, it fails for other words due to the following challenges.



(a) (b)

Figure 2. (a) Actual finger strokes for entering *work: w-r* and *o-k.* (b) Final gesture by connecting all touch points according to timestamps: *w-o-w-r-o-k.* Blue strokes are actual finger strokes.

First, unimanual gesture recognition algorithms are based on the assumption that the gesture travels through letters of a word following their natural order. For example, the unimanual gesture template for *work* is *w-o-r-k*. The unimanual gesture algorithm gives the matching score between a unimanual gesture and this template. However, bimanual gestures do not always follow this rule. When a finger slides to a target letter, it may move from the last letter in the word *on the same side*, which might not be the letter preceding the target letter in the word.

For example, *one way* to enter *work* on a bimanual gesture keyboard is as follows (Figure 2):

1. Left thumb presses the letter *w* and right thumb presses *o*. Connecting touch points based on the timestamps creates the stroke *w-o*.
2. Left thumb slides to *r* from its previous location *w*. Connecting all the touch points together creates the stroke *w-o-w-r*.

3. Right thumb slides to k from o. The stroke becomes *w-o-w-r-o-k*, which is different from the unimanual gesture template, *w-o-r-k*.

Second, users might gesture the same word differently, depending on whether the finger slides from its previous location, or starts over from a new location. This variance leads to different gestures for the same word. For example, *interaction* can be gestured in the following two approaches:

Approach #1 (Figure 3a):

1. Right finger gestures *i-n;*
2. Left finger gestures *t-e-r-a-c-t;*
3. Lift right finger from the previous letter *n,* land it on the letter *i,* and gesture *i-o-n;*

The final gesture according to the timestamp is *i-n-t-e-r-a-c-t-i-o-n*. This one is the same with the unimanual gesture template.

Approach #2 (Figure 3b):

1. Right finger gestures *i-n;*
2. Left finger gestures *t-e-r-a-c-t;*
3. Slide right finger to *i* from its previous location *n*, and gesture *i-o-n*.

The final gesture becomes *i-n-t-e-r-a-c-t-**n**-i-o-n,* different from the unimanual gesture template.

(a) (b)

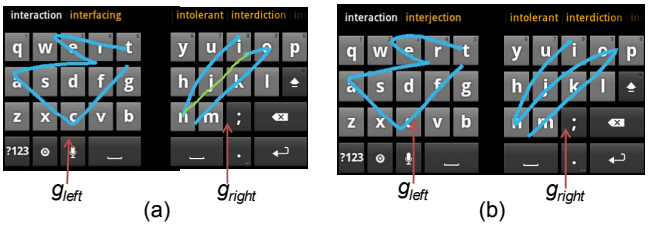
Figure 3. Two approaches for entering *interaction.* (a). Approach #1, (b) Approach #2.

Third, users might move two fingers simultaneously for a given word. For example (Figure 1), *life* is usually gestured by *l-i* (right finger), followed by *f-e* (left finger). Once a user gets familiar with the pattern, she might start the second stroke *f-e* before finishing *l-i*. Therefore, some touch points on the stroke *f-e* have earlier timestamps than those on *l-i*. Simply connecting touch points based on timestamps produces a gesture different from the unimanual gesture template *l-i-f-e*.

To address the above challenges, we designed a novel *multi-stroke gesture recognition algorithm.* Given a bimanual gesture *g*, it generates a ranked list of candidate words.

It works as follows:

Step 1. Reorganizing touch points based on finger id. Touch points on the gesture *g* with the same finger id are connected together to generate a sub-gesture based on the timestamps. On a split keyboard, touch points on the same side are connected together, since they are generated by the same finger. This step leads two sub-gestures: g_{left} and g_{right}, with the finger id, *left* and *right* respectively (Figure 4).



(a) (b)

Figure 4. Generating g_{left} and g_{right} by connecting touch points with the same finger id together. (a) and (b) show two approaches for entering the word *interaction.* Blue strokes represent actual finger strokes. The green line in (a) connects the last point of stroke *i-n* with the first point of *i-o-n*, because both of them were drawn by right finger. As shown, (a) and (b) lead to similar sub-gestures.

Step 2. Labeling each letter of a given word *w* with the id of the finger that gestures it, and splitting the word into sub-strings based on finger ids. On a split keyboard, letters on the left and right sides are gestured by left and right fingers respectively. The finger id of a letter is determined based on its location. The word *w* is then split into two sub-strings, $s_{left}(w)$ and $s_{right}(w)$, with the finger id *left* and *right* respectively. Take the word *interaction* as an example. $s_{left}(w)$ = *teract,* and $s_{right}(w)$ = *inion,* because letters *i, n, i, o, n* are gestured by right finger while the others by left finger.

Step 3. Matching sub-gestures g_{left} and g_{right} with sub-strings $s_{left}(w)$and $s_{right}(w)$ respectively. The match scores are recorded as $c_{left}(w)$ and $c_{right}(w)$, reflecting the probabilities of s_{left} and s_{right}, given the g_{left} and g_{right}, respectively. A unimanual gesture recognizer[1] is used here to calculate the matching score between a sub-string and a sub-gesture. E.g., if *w* = *interaction*, $g_{left}(w)$ and $g_{right}(w)$ (Figure 4) are matched with $s_{left}(w)$ = *teract* and $s_{right}(w)$ = *inion*, respectively. The unimanual gesture recognizer first searches for the closest touch point along a sub-gesture for a given letter. The distance between this touch point and the letter is converted to a distance score. The sum of distance scores over all the letters within a word is mapped to the matching score.

Step 4. Obtain the gesture match score *c(w)* for a given word *w* by integrating $c_{left}(w)$ and $c_{right}(w)$:

[1] A detailed description of the unimanual gesture recognition algorithm is beyond the scope of this work, and is independent of the bimanual gesture issues investigated in this paper. We used algorithms different from what has been published in the literature such as Kristensson and Zhai [13], but that approach could also be adapted as building blocks to bimanual gesture algorithm. The algorithms for other unimanual gesture keyboards such as Swype, SlideIt, T9 trace, Touchpal have not been published in the scientific literature, but any of them may be applied here as well.

$$c(w) = \frac{c_{left}(w)c_{right}(w)}{\sum_{i \in W} c_{left}(i)c_{right}(i)}$$

where W is the set of the words contained in the N-best list. N-best list is the set containing the words with top $c_{left}(w)c_{right}(w)$ values. $N = 500$ in our implementation.

Step5. Obtain the final score $c'(w)$ by integrating gesture score $c(w)$ with the language model score $l(w)$:

$$c'(w) = \frac{c(w)l(w)}{\sum_{i \in W} c(i)l(i)}$$

Where $l(w)$ is the probability of the word w based on language model, and W is the set of words contained in the N-best list. The language model in the experiment contained approximately 90,000 unigrams and 300,000 bigrams.

The words in the N-best list are ranked according to the confidence score $c'(w)$. The top candidate is displayed directly in the text input view. Other candidates are displayed on the suggestion bar.

In sum, by reorganizing touch points and splitting letters based on finger id, this algorithm reduces the bimanual gesture recognition problem to two unimanual sub-gesture recognition problems. Since g_{left} and g_{right} are processed separately and independently, it supports both sequential and concurrent finger movements.

Implementation of Bimanual Gesture Keyboards

Based on the aforementioned methods and algorithm, we developed *finger-release* and *space-required* bimanual gesture keyboards on Android Devices. Figure 5 illustrates their architectures.

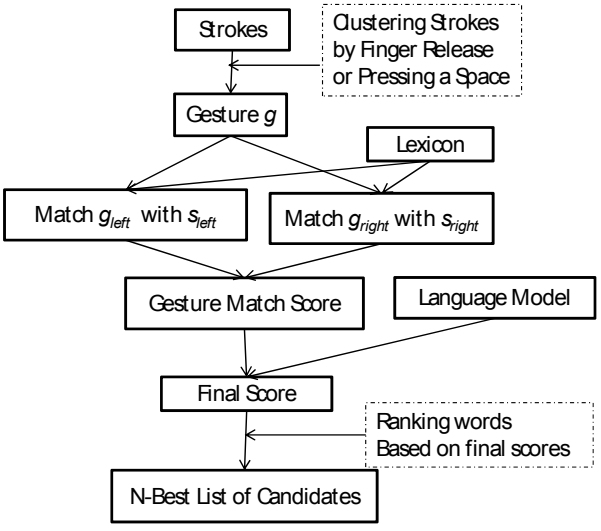


Figure 5. The architecture for bimanual gesture keyboards.

EMPIRICAL STUDY

Our bimanual gesture algorithms enable a new type of interaction behavior that was previously not possible, and we further analyzed and evaluated this design with an empirical study.

The goal of this study was twofold. First it closed the design-iteration loop by testing with participants who had no knowledge of, or bias from, the research and design insight. We wanted to observe users' actual behaviors on bimanual gesture keyboards and evaluate the strengths and weaknesses of two bimanual gesture keyboard methods.

Second, we evaluated bimanual gesture performance along with unimanual gesture performance. Although gestures have already been adopted on a large number of devices, formal user studies of unimanual gesture keyboards are still very limited in the literature.

Note that our basic goal is to expand the flexibility of the gesture keyboard paradigm. If the expanded bimanual mode is completely compatible with existing input modes and is preferred by a meaningful number of users, it does not need to "beat" other modes in performance to be valuable. However it is still important and informative to measure bimanual gestures movement and time difference in comparison to their unimanual counterparts in order to gain empirical insights.

Although the bimanual gesture keyboards work on both smart phones and tablets, this study focused on tablets in order to maintain a manageable scope.

Design

We used a 2 × 3 × 2 mixed-factorial design, with one between subject factor, tablet orientation (Portrait and Landscape), and two within-subject factors: input method and input tasks (phrase vs.word). Input method had three levels: unimanual gesture, finger-release and space-required bimanual gestures.

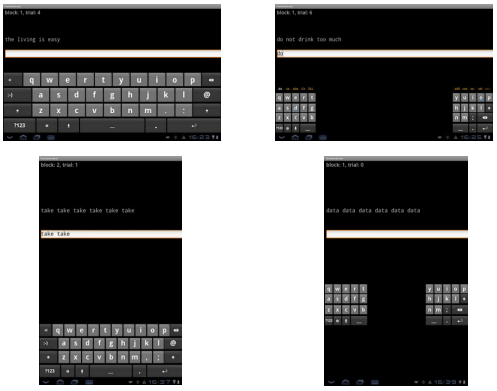


Figure 6. Left: unimanual gesture keyboards. Right: bimanual gesture keyboards.

Figure 6 illustrates the keyboards used for each method. We chose the sizes based on practical usages. The sizes (in pixels) of unimanual gesture keyboards were 1280 × 360 with each letter key 105 × 90 in landscape mode, and 800 × 304 with each letter key 66 × 76 in portrait. These were also default sizes for Android keyboards on a tablet. The sizes of bimanual gesture keyboards were identical in both landscape or portrait modes: 240 × 240 on each side, with the letter key size 48 × 60. These values were chosen to fit the two hand holding postures. Input task included two levels: *phrase input* and *word repetition* [5,6]. The former

simulated regular text entry tasks, while the latter investigated the learnability and expert input speed of each input method. The main measures were *Input Speed* (WPM), *Not Corrected Error Rate*, and *Corrected Error Rate* [22].

Phrase input. In each trial, the participant entered a phrase chosen from MacKenzie and Soukoreff's phrase set [18]. To randomize the starting position of the input finger and also record the beginning time of a trial, participants started each trial by tapping a red start circle (diameter = 110 pixels) on the unimanual keyboard, and tapped the *Enter* key after finishing the phrase. The start circle was centered on one of the 26 English letters. The probability of appearing on a given letter was in accordance with its frequency as an ending letter in English Word, estimated from the American National Corpus.

There were two start circles on a bimanual keyboard, to randomize the positions of two fingers.

Participants were allowed to correct the current word by backspace key.

The input speed was calculated as:

$$WPM = \frac{|T|}{S} \times 60 \times \frac{1}{5}$$

where T was the final transcribed string and S was the elapsed time in seconds from the moment when the finger was lift from staring circle to the finish of the last word in the phrase. Note that the numerator was $|T|$, instead of $|T| - 1$ [23] because the time to input the first character was also included.

The error rates were:

$$\textit{Not Corrected Error Rate} = \frac{INF}{C + INF + IF}$$

$$\textit{Corrected Error Rate} = \frac{IF}{C + INF + IF}$$

where C is the total number of correct words, IF is the number of incorrect but fixed(backspaced) words, and INF is the number of incorrect (but not fixed) words. We used a word-level instead of a character-level error rate [22] because both gesture keyboards were word-level input methods. We randomly chose 32 phrases from MacKenzie and Soukoreff's phrases [18] as the test set and divided them into 4 blocks. The orders of phrases were randomized within a block. The first block was a warm-up session.

Word Repetition. In each trial, the participant entered the same word six times repeatedly. Examining how input speed progressed as a word was repeatedly entered revealed the learnability of a method. Also, assuming that a user became familiar with the gesture pattern of the word after repeating it multiple times, the last one or two repetitions simulated the expert input speed a user can achieve [5, 6].

In each trial, the participant started each repetition by pressing the red starting circle(s). The finish of a word was determined as follows:

1) In unimanual and *finger-release* bimanual gesture conditions, the target word was correctly gestured, or picked from the suggestion bar, or the participant started the next repetition without correcting the current word which was incorrectly gestured.
2) In space-required bimanual, a space key was tapped, or the word was picked from the suggestion bar.

New start circles appear upon the finish of a word.

The input speed was calculated as:

$$WPM = \frac{|W|+1}{S} \times 60 \times \frac{1}{5}$$

where $|W|+1$ was the length of the target word plus a space character and S was the elapsed time in seconds from the moment when participants lifted fingers from the starting circles to the finish of a word. The error rates were the same with those in *Phrase Input.*

The test set included 32 words randomly chosen from the top 300 frequently used English words. These words were equally divided into 4 blocks and the order within a block was randomized. The first block was a warm-up session.

Subjective measures were collected at the end of the experiments. Participants rated each input method based on *comfort* , *learnability*, *efficiency*, *accuracy*, *absence of frustration*, *mental, physical demands*, and *overall preference*, all on 1~10 scales. Some of the measures were adopted from NASA TLX [10].

We also recorded detailed information of each touch event, to analyze participants' gesturing and tapping behaviors.

Participants

We recruited 36 participants (17 females) between ages of 18 and 56. All of them used tablets at least several times a month, 17 used it daily. Eighteen of them participated in the Portrait condition while the others in Landscape. The orders of the three input methods and two tasks were fully counterbalanced across participants. Each participant entered 3 × 32 = 96 phrases in *phrase input*, and 3 × 6 × 32 = 576 words in *word repetition* tasks. The study lasted around 1.5 hours for each participant.

Apparatus

The experiment was conducted on a Samsung Galaxy Tab 10.1 with Android OS 2.3. The size of the multi-touch screen was 256.7 × 175.3 mm with resolution of 1280 × 800. The weight was 565g. Participants held the tablet with either one or two hands while sitting in a chair (Figure 7).

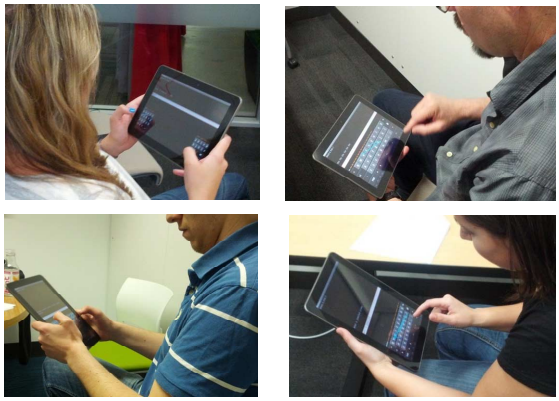


Figure 7. Participants held tablets with two hands (left) or one hand (right).

Results

We first investigated the effect of input method on *input speed* in *phrase input,* and the average input speed of last two repetitions in *word repetition*, which simulated the expert input speed. Similar effects were discovered for both of them. (Figure 8). Input method had a significant main effect on both measures ($F_{(2, 68)}$ = 87.562, for phrase input, $F_{(2, 68)}$ = 153.1,for last two repetitions; $p < 0.01$). Pairwise comparisons showed significant differences between every two methods ($p < 0.01$). ANOVA did not show a main effect of orientation, or orientation × input method interaction.

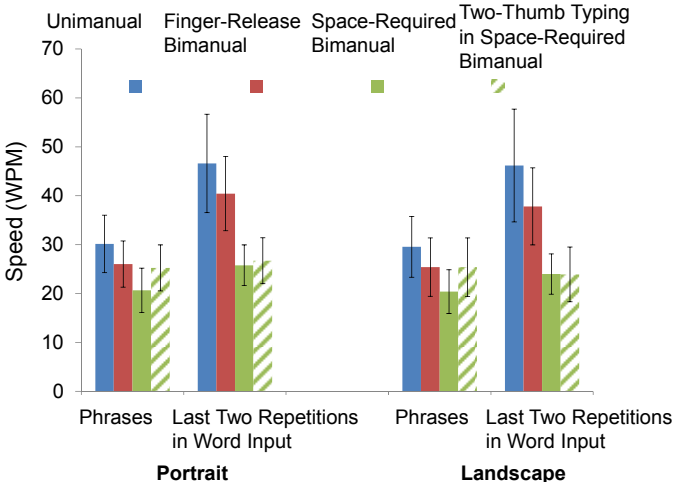


Figure 8. Mean (SD) input speed.

To understand the learnability of each method, we investigate the effect of repetition position (#1~6) on input speed, for each input method separately. The analysis showed similar results across all three methods (Figure 9): participants reached a relatively stable speed from repetition #2. ANOVAs showed that word position had significant main effects on speed ($F_{(5, 170)}$ = 27.168 for *unimanual*, $F_{(5, 170)}$ = 20.443 for *finger-release bimanual*, $F_{(5, 170)}$= 39.671 for *space-required bimanual*, $p < 0.01$ for all.), pairwise mean comparisons showed significant differences between position #1 with any of other positions ($p < 0.05$), but not between any other two positions. The analysis did not show any significant main effect of orientation on speed.

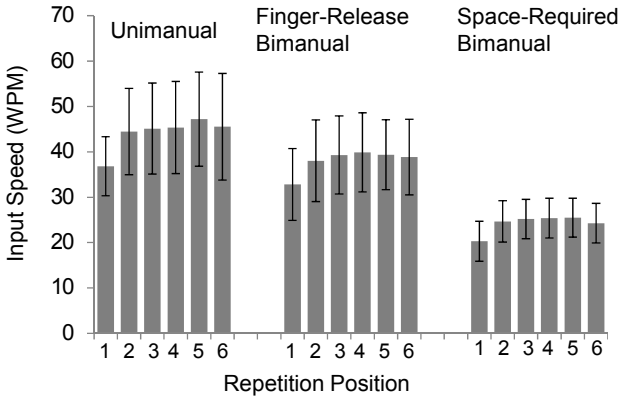


Figure 9. Mean(SD) input speed in Word Repetition Tasks.

We did not explicitly include two-thumb typing as an experimental condition given the limited manageable scope of one controlled study. However, *space-required* bimanual gesture can be viewed as enhanced two-thumb typing: users can tap a word or mix tapping with gesturing. Examining the subset of words that were exclusively entered by tapping provides some insights about two-thumb typing.

On average, 52% (SD = 15%) of words in *phrase input* and 31% (SD = 9%) in *word repetition* were entered by tapping only. Tapping and gesturing were identified by stroke length. A stroke with the length less than half a key-width was identified as a tap. As shown in Figure 8, input speed of *finger-release* bimanual gesture was close to that of two-thumb typing in *phrase input*, but around 50% faster for expert input speed (i.e., last two repetitions in word repetition task). Since these words represent just a subset of the full experimental dataset, we did not perform inferential statistical analysis on them.

Not Corrected Error Rate

ANOVA did not show any significant main effect of orientation ($F_{(1, 34)}$ = 0.257, p = 0.615) or input method ($F_{(2, 68)}$ = 0.522, p = 0.595) on *Not Corrected Error Rate*, indicating that participants input texts at the same level of accuracy in both portrait and landscape modes, and across different input methods. The mean (SD) were unimanual at 2.8% (3.2%), *finger-release* bimanual at 3.3% (3.6%), *space-required* bimanual at 3.9% (5.0%). The task type had a significant main effect ($F_{(1, 34)}$ = 13.53, $p < 0.05$) with the mean (SD) 4.1%(4.8%), 2.5%(3.0%) for *phrase input* and *word repetition* respectively.

Corrected Error Rate

Unlike Not Corrected Error Rate, Input method has a significant main effect on Corrected Error Rate, i.e., the percentage of words that were backspaced ($F_{(2, 68)}$ = 10.276, p <0.001), with the means (SD) of 5.5% (3.1%) for unimanual, 7.6%(4.9%) for *finger-release* bimanual, and 8.1%(5.6%) for *space-required* bimanual. Pairwise mean comparisons show significant differences for *unimanual* vs.

finger-release bimanual, and *unimanual* vs. *space-required bimanual* ($p < 0.05$). ANOVA did not show any significant main effect for either task type ($F_{(1, 34)} = 1.950$, $p = 0.172$), or orientation ($F_{(1, 34)} = 0.252$, $p = 0.619$).

Gesture length

To investigate the movement efficiency of each method, we examined the gesture length per word, i.e., the distance the finger(s) travel on the screen for a given word. ANOVA showed that orientation ($F_{(1, 34)} = 4790$, $p < 0.01$), task type ($F_{(1, 34)} = 36.4$, $p < 0.01$) and input method ($F_{(2, 68)} = 3184$, $p < 0.01$) had significant main effects on gesture length. Pairwise mean comparisons showed significant differences between every two input methods ($p < 0.05$).

As shown in Figure 10, in portrait mode, the gesture length of unimanual gesture was 3.5 and 6.2 times longer than that in *finger release*, and *space-required* bimanual, respectively. In landscape mode, it was 2.1, and 3.4 times, respectively.

Subjective Measures

Participants rated each input method based on the level of *comfort*, *learnability*, *efficiency*, *accuracy*, *absence of frustration*, *mental, physical demands*, and *overall preference* in 1-10 scales. Results are illustrated in Figure 11. We performed ANOVAs, as well as post hoc pairwise comparisons with Bonferroni adjustment on each measure (the normality of ratings in each measure passed the Shapiro-Wilk tests). Finger-release and space-required bimanual were rated similarly across all the measures. No significant difference ($p > 0.05$) was observed in any measure.

Unimanual was rated most positive from measure 2 to 6, but most negative in 1 (*comfort*) and 7 (*physical demand*), probably due to the long finger travel distance and the fatigue caused by holding the tablet with one hand. The differences between *unimanual* and other method in both measures 1 and 7 were significant ($p < 0.05$).

The means (SD) of overall preferences rating were *unimanual* at 7.9 (2.3), *finger-release bimanual* at 6.6 (2.7), *space-required bimanual* at 5.7 (2.7). The difference for *unimanual vs. space-required* is significant ($p < 0.05$), but not between every other two methods. Participants were asked to choose one method from the three gesture keyboards and the regular two-thumb typing as the most preferred one for tablets. 19 chose unimanual, 11 chose finger-release bimanual, 4 chose space-required bimanual and 2 chose two-thumb typing.

DISCUSSION

Strengths and Weaknesses of Bimanual Gesture Keyboards

Learnability. Our study results showed that both *finger-release* and *space-required* bimanual gestures were easy to learn. The mean (SD) subjective measure regarding learnability (Figure 11) were 6.3 (2.2) and 7.0 (2.0) for *finger-leased* and *space-required* bimanual respectively, indicating the high learnability for both of the two methods. Results also showed that *finger-release* was at the similar level of learnability with *space-required* bimanual method. Also, bimanual gestures can be quickly adopted by users. As shown in Figure 9, participants quickly improved the input speed by repeating the word once.

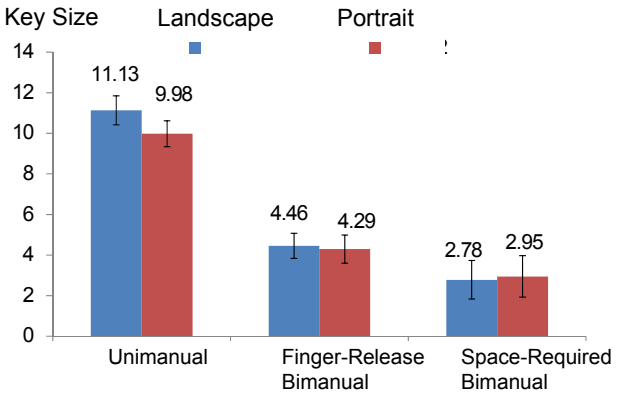


Figure 10. Mean (SD) gesture length per word. Key Size = (Key Width + Key Height) / 2.

Movement Efficiency. Since each thumb only moves in its vicinity on one side of the keyboard, we conjecture that bimanual gesture keyboards have higher movement efficiency over unimanual gesture keyboards. To better understand it, we compared the average length of gesture templates on different keyboards. The gesture template of a word is a set of line segments connecting centers of keys, following the order of letters within the word. For a word with multiple gesture templates on a bimanual gesture keyboard (e.g., Figure 3, two approaches for entering *interaction*). We chose the one with the longest gesture length, to approximate the upper bound.

The average lengths of gesture templates over all words in English Corpus [25], weighted with word frequency, are 11.6 and 5.32 key size long on unimanual and bimanual gesture keyboards respectively, predicting that bimanual gesture keyboard shortens the finger travelling distance by approximately 50% over its unimanual counterpart. The empirical data (Figure 10) echoes this finding: the finger release bimanual gesture keyboard shortens the gesture lengths by 60% (landscape) and 57% (portrait) over the unimanual gesture keyboard.

The shorter travelling distance reduces the physical demands of entering words, and raises the comfort levels (Figure 12). Some participants reported they did not like unimanual gesture on a tablet, especially in landscape mode, because the long distance travelling causes too much friction. Some users even tilted fingers to reduce the friction when drawing unimanual gestures across the screen (Figure 12).

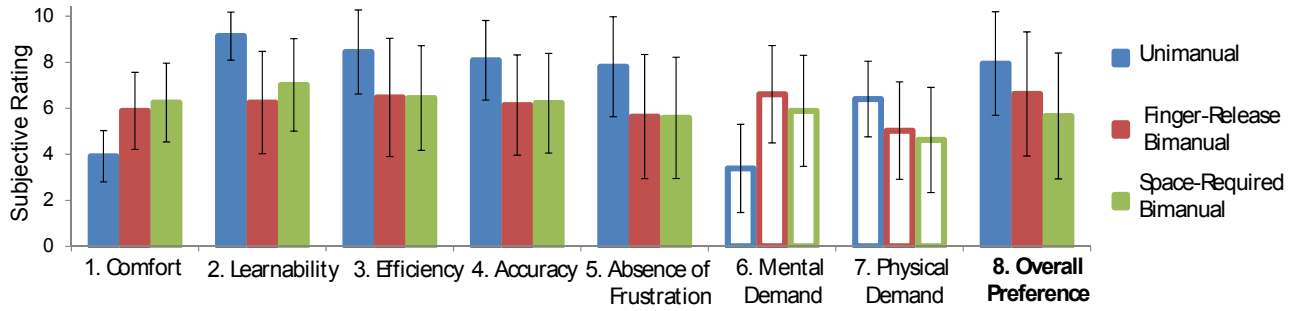


Figure 11. Mean (SD) of subjective ratings, all in (1-10) scales. For measures 1-5, and 8, a score of 10 is the most positive rating. For measures 6 and 7, 10 is the most negative rating (highest demand).

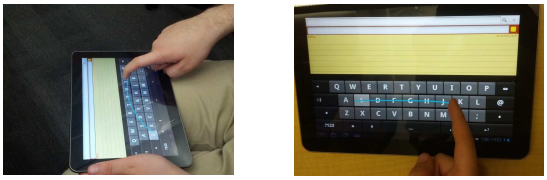


Figure 12. Users tilted fingers to reduce friction on unimanual gesture keyboards.

Time Efficiency. *Finger-release* is faster than *space-required* bimanual, mainly because it omits the space input. Its phrase input and the expert speed were 26, and 39 WPM respectively. The expert input speed is also around 50% faster than that of two-thumb typing in *space-required* bimanual (Figure 8).

Surprisingly, the movement efficiency of bimanual gesture keyboard does not lead to higher time efficiency over a unimanual gesture keyboard. It was slightly slower than unimanual counterpart, with the margins of 4 and 7 WPM for phrase input and expert speed, respectively. One possible reason is that the attention and action switches between two fingers were costly. They slowed users down and broke the interaction flow. Bimanual gesture might require more effort to coordinate two thumbs. Subjective measures showed that both bimanual gesture methods required more mental effort than bimanual gestures. Participants also corrected the input words more often on bimanual gesture keyboards than on the unimanual gesture keyboard, which reduced the time efficiency.

Screen Space Consumption. As a full keyboard is split to two sides, bimanual gesture keyboards consumed much smaller screen real estate than the unimanual gesture keyboards, which was important to boost the overall user experience on portable touchscreens [15]. In current implementations, the sizes of bimanual keyboards were 240 × 240 on each side (Figure 6), which were just 25% (Landscape), and 47% (Portrait) of unimanual gesture keyboards.

No Keyboard Fits All

Users' preferences varied when they selected the most preferred method. Unimanual gesture keyboard was the fastest, but it did not have the dominance. Only 19 participants, around 50% of all the subjects, select it as the most favorite input method. One disadvantage is that it has high physical demand and lower level of comfort in mobile situations. One female participant explicitly mentioned that the tablet was too heavy to hold with one hand. Also it occupies more screen space then split keyboards, and does not fit the two hand holding, a common posture in mobile situations.

Eleven participants, around 31% of all the subjects, chose finger-release bimanual as the most preferred input method. Four of them entered texts fastest with this method. Other participants mentioned that it well fitted the two-hand holding posture, and omitted the space input. Some participants also reported that they preferred bimanual over unimanual gesture keyboards because they consumed much smaller screen space.

Four participants preferred the space-required bimanual method, even though the inputting speed is slower than others. They reported that it was similar to regular tapping, and they loved the flexibility of mixing gesturing with tapping. They also reported that they did not intensively enter text on tablets, at most a few minutes each time for replying messages. The speed was not a critical factor. Other two Participants selected the two-thumb typing as the most preferred, mainly because of its familiarity.

One important lesson learned from the study was that no one input method fits all the users. Many factors came into play when choosing the input method, and these factors were weighted differently across users. Some users favor speed, while others might consider the ergonomics, familiarity, or the overall balance across different factors.

CONCLUSION

This paper presents a new interaction behavior, bimanual gesture text entry, which expands the gesture keyboard from one finger to multiple fingers.

Recognizing bimanual gesture is challenging and more complicated than recognizing unimanual gesture, because the number of strokes per word can vary, users can draw gestures differently for the same word, and gestures may overlap in time. To address these challenges, we have proposed two interaction methods, *finger-release* and *space-required*, and described the multi-stroke gesture recognition algorithm for recognizing both methods. Furthermore, we have designed and implemented a bimanual gesture text entry system that supports both operation modes. Both are compatible with unimanual gesture and regular typing.

We also conducted an empirical study to evaluate the strengths and weaknesses of bimanual gesture keyboards.

For phrase entry, the average speed of the *finger-release* and *space-required* techniques were 26 WPM and 21 WPM, respectively. For the last two repetitions of the word repetition exercise, which simulated expert performance, speeds increased to an average of 39 WPM and 26 WPM. Although both techniques were slower than unimanual gesture (which averaged 30 WPM for phrases and 46 WPM for word repetition), bimanual stroke lengths were shorter and more efficient. Participants also subjectively reported that bimanual systems were more comfortable and less physically demanding. Fifteen out of thirty-six participants (42%) favored the bimanual gesture keyboards over unimanual. Overall, these results indicate that bimanual gesture keyboards are valuable complements to existing text entry methods.

REFERENCES

1. Anthony, L. and Wobbrock, J.O. (2010). A lightweight multistroke recognizer for user interface prototypes. *Graphics Interface* 245-252.
2. Balakrishnan, R., and Hinckley, K. (2000). Symmetric bimanual interaction. *ACM CHI.* 33-40
3. Balakrishnan, R., and Kurtenbach, G. (1999). Exploring bimanual camera control and object manipulation in 3D graphics interfaces. *ACM CHI.* 56-63.
4. Bier, E. A., Stone, M., Pier, K., Buxton, W. & DeRose. T. (1993). Toolglass and magic lenses: the see-through interface. *SIGGRAPH*, 73-80.
5. Bi, X., Smith, B., and Zhai, S. (2010). Quasi-qwerty soft keyboard optimization. *ACM CHI.* 283-286.
6. Bi, X., Smith, B., and Zhai, S. (2012) Multilingual Touchscreen Keyboard Design and Optimization, *Human Computer Interaction*, to appear. Available online.
7. Buxton, W., and Myers, B. (1986). A study in two-handed input. *ACM CHI*, 321-326.
8. Findlater, L., Wobbrock, O., Wigdor, D. (2011). Typing on flat glass: examining ten-finger expert typing patterns on touch surfaces. *ACM CHI* 2453-2462.
9. Guiard, Y. (1987). Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model. *Journal of Motor Behavior*, 19(4), pp. 486-517.
10. Hart, S. and Staveland, L. 1988. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Human Mental Workload P. A. Hancock and N. Meshkati, Eds.,* North-Holland: Elsevier Science, 139-183.
11. Kane, S.K., Wobbrock, J.O. and Ladner, R.E. (2011) Usable gestures for blind people: Understanding preference and performance. *ACM CHI*. 413~422
12. Kristensson, P.O,. Zhai, S. (2007) Command Strokes with and without Preview: Using Pen Gestures on Keyboard for Command Selection, *ACM CHI.* 1137~1146
13. Kristensson, P-O., Zhai, S., S (2004) SHARK^2: A Large Vocabulary Shorthand Writing System for Pen-based Computers, *ACM UIST,* 43–52.
14. Kristensson, P-O., Zhai, S. (2005) Relaxing Stylus Typing Precision by Geometric Pattern Matching. *ACM IUI* 151–158.
15. Li, F.C.Y, Guy, R.T., Yatani, K., Truong, K.N. (2011). The 1line keyboard: a QWERTY layout in a single line. *ACM UIST*, 461-470.
16. Li, Y. (2010). Protractor: A Fast and Accurate Gesture Recognizer, *ACM CHI,* 2169~2172
17. Li, Y. (2010). Gesture Search: A Tool for Fast Mobile Data Access *ACM UIST*, 87~96
18. MacKenzie, I.S., Soukoreff, R. W. (2003). Phrase sets for evaluating text entry techniques. *Extended Abstracts CHI*, 754-755.
19. Ouyang, T., and Davis, R. (2009). A Visual Approach to Sketched Symbol Recognition, *IJCAI*, 1463~1468
20. Ouyang, T., and Li, Y. (2012), Bootstrapping Personal Gesture Shortcuts with the Wisdom of the Crowd and Handwriting Recognition, *ACM CHI*, to appear.
21. Rick, J. (2010). Performance optimizations of virtual keyboards for stroke-based text entry on a touch-based tabletop. *ACM UIST,* 77-86.
22. Soukoreff, R.W., MacKenzie, I.S. (2003) Metrics for text entry research: an evaluation of MSD and KSPC, and a new unified error metric. *ACM CHI*, 113-120.
23. Wobbrock, J.O. (2007) Measures of text entry performance. In *Text Entry Systems: Mobility, Accessibility, Universality*, I. S. MacKenzie and K. Tanaka-Ishii (eds.) San Francisco: Morgan Kaufmann, 47-74
24. Wobbrock, J.O., Wilson, A.D. and Li, Y. (2007). Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. *ACM UIST* 159-168.
25. www.americannationalcorpus.org/
26. Zhai, S. Kristensson, P.O. Introduction to Shape Writing, *IBM Research Report* RJ10393 (A0611-006), November1, 2006
27. Zhai, S., Kristensson, P-O., (2003) Shorthand Writing on Stylus Keyboard, *ACM CHI.* 97-104.
28. Zhai, S., Kristensson, P.O., Gong, P., Greiner, M., Peng, S., Liu, L. Dunnigan, A., (2009) Shapewriter on the iPhone: from the laboratory to the real world. *ACM CHI Extended Abstracts*. 2667-2670.