The Feasibility of Eyes-Free Touchscreen Keyboard Typing

Keith Vertanen
Montana Tech
Butte, Montana, USA
kvertanen@mtech.edu

Haythem Memmi
Montana Tech
Butte, Montana, USA
hbmemmi@mtech.edu

Per Ola Kristensson
University of St Andrews
St Andrews, UK
pok@st-andrews.ac.uk

ABSTRACT

Typing on a touchscreen keyboard is very difficult without being able to see the keyboard. We propose a new approach in which users imagine a QWERTY keyboard somewhere on the device and tap out an entire sentence without any visual reference to the keyboard and without intermediate feedback about the letters or words typed. To demonstrate the feasibility of our approach, we developed an algorithm that decodes blind touchscreen typing with a character error rate of 18.5%. Our decoder currently uses three components: a model of the keyboard topology and tap variability, a point transformation algorithm, and a long-span statistical language model. Our initial results demonstrate that our proposed method provides fast entry rates and promising error rates. On one-third of the sentences, novices' highly noisy input was successfully decoded with no errors.

Categories and Subject Descriptors

K.4.2 [**Computers and Society**]: Social Issues - assistive technologies for persons with disabilities.

1. MOTIVATION AND APPROACH

Entering text on a touchscreen mobile device typically involves visually-guided tapping on a QWERTY keyboard. For users who are blind, visually-impaired, or using a device eyes-free, such visually-guided tapping is difficult or impossible. Existing approaches are slow (e.g. the split-tapping method of the iPhone's VoiceOver feature), require chorded Braille input (e.g. Perkinput [1], BrailleTouch [3]), or require word-at-a-time confirmation and correction (e.g. the Fleksy iPhone/Android app by Syntellia).

Rather than designing a letter- or word-at-a-time recognition interface, we present initial results on an approach in which recognition is postponed until an entire sentence of noisy tap data is collected. This may improve users' efficiency by avoiding the distraction of intermediate letter- or word-level recognition results. Users enter a whole sequence of taps on a keyboard they imagine somewhere on the screen but cannot actually see. We then decode the user's entire intended sentence from the imprecise tap data. Our recognizer searches for the most likely character sequence under a probabilistic keyboard and language model.

Figure 1: Test development interface. Shown are the taps and recognition results before (left) and after transformation (right). Taps were scaled horizontally and slightly translated/rotated. Taps are colored from red (first) to blue (last). The user tapped "have a good evening" without a visible keyboard.

The keyboard model places a 2D Gaussian with a diagonal covariance matrix on each key. For each tap, the model produces a likelihood for each of the possible letters on the keyboard with higher likelihoods for letters closer to the tap's location. Our 9-gram character language model uses Witten-Bell smoothing and was trained on billions of words of Twitter, Usenet and blog data. The language model has 9.8 M parameters and a compressed disk size of 67 MB.

Since users are imagining the keyboard's location and size, their actual tap locations are unlikely to correspond well with any fixed keyboard location. We compensate for this by geometrically transforming the tap points as shown in Figure 1. We allow taps to be scaled along the x- and y-dimensions, translated horizontally and vertically, and rotated by up to 20 degrees. We also search for two multiplicative factors that adjust the x- and y-variance of the 2D Gaussians.

Our current decoder operates offline, finding the best transform via a grid search. Transforms are ranked by first transforming a tap sequence and then making a *fixed* decoding pass. The pass is fixed in that we make a greedy decision for the best letter for each tap, fixing our decision for the rest of the search. This allows us to quickly evaluate many possible transforms. The probability of the resulting char-

Table 1: Error rates from our first experiment.

Model	CER	WER	SER
No transform	60.5	83.0	97.0
Transform	35.4	56.7	84.5
Transform + variances	36.6	52.7	80.0
Combination	**32.9**	**49.1**	**77.8**

acter sequence is taken as the score for a transform. Using the highest scoring transform we then perform a *full* decoding pass. In full decoding, all character sequences are potentially considered. To make the search tractable, we use beam width pruning to focus the search.

2. DATA COLLECTION AND RESULTS

We developed an iPhone app that collected tap data. Users heard an audio recording of a short stimulus sentence whenever they touched the screen with two fingers at the same time. To simulate not being able to see the keyboard, we blindfolded users. The app merely recorded tap positions, no recognition was performed on the device.

We measured entry rate in words per minute (wpm). A word was defined as five characters (including spaces). Time was measured from a sentence's first tap until a double-touch. Error rate was measured using character error rate (CER). CER is the number of characters that must be substituted, inserted or deleted to transform the user's entry into the stimulus, divided by the length of the stimulus. We also report word error rate (WER), which is analogous but on a word-basis, and sentence error rate (SER), which is the percentage of sentences that had one or more errors.

In our first experiment, 14 participants entered 20 sentences chosen at random from short memorable sentences from the Enron mobile test set [4]. All participants were familiar with the QWERTY keyboard. Other than the playback of sentences, no audio or tactile feedback was provided. There was no mechanism to correct errors. Participants were told to hit an imaginary spacebar between words.

Participants' mean entry rate was 29.4 wpm. Table 1 shows the error rates for different approaches: full decoding without transformation, full decoding with transformation, and full decoding with transformation and keyboard variance optimization. Combining all models improved accuracy. Combination was performed by choosing the model result that was most probable under the language model.

Given the error rates in our first experiment, we realized we needed more signal from users. We did this by modifying our app to require a right swipe gesture for spaces (similar to [2]). We classified a touch event as a swipe if its width was over 52 pixels. Our decoder was modified to only insert spaces for swipe events. We also added audio feedback. For taps we played the standard iPhone keyboard click sound. For swipes we played the standard iPhone unlock sound.

In our second experiment, 8 participants entered 40 sentences while blindfolded. All participants were familiar with the QWERTY keyboard. Participants' mean entry rate was 23.3 wpm. Table 2 shows the error rates using different transforms. Since we had information about word boundaries based on the right swipe, we also tested computing geometric transforms for each word independently. We conjectured this might help if users' imagined keyboard location or size drifted between words. The right swipe gesture made recognition much easier. Independent word transforms did worse on average than a single sentence transform but did help improve accuracy when combined with other models.

Table 2: Error rates from our second experiment.

Model	CER	WER	SER
No transform	51.1	80.4	90.9
Transform	20.0	32.2	67.2
Transform + variances	27.0	41.4	74.5
Word transform	25.1	40.9	80.5
Word transform + variances	31.0	49.0	86.1
Combination	**18.5**	**30.1**	**67.9**

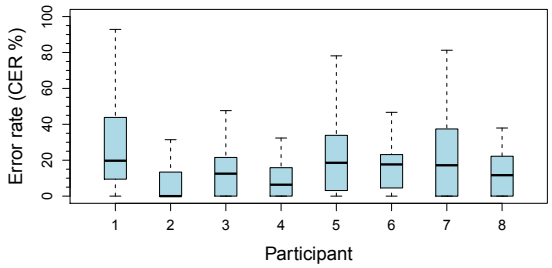

Figure 2: Error rates from our second experiment.

As shown in Figure 2, individual error rates were variable. Our best user had an error rate of 9.8%. Exactly why this user had such a relatively low error rate is unknown. But it is plausible this participant was more careful and accurate in tapping. This provides hope that, at least with practice, users may eventually achieve much lower error rates.

3. CONCLUSIONS

We have proposed a new approach to touchscreen keyboard typing in which users imagine a keyboard somewhere on the device and tap out an entire sentence without any visual reference to the keyboard. Our preliminary results show this may be a viable approach. While error rates are still somewhat high, there remain numerous avenues for improvement. Future work includes: a) improving recognition accuracy, b) implementing efficient error correction interfaces, c) investigating how to obtain a better signal from users, and d) collecting data from users who are blind or visually-impaired.

4. REFERENCES

[1] S. Azenkot, J. O. Wobbrock, S. Prasain, and R. E. Ladner. Input finger detection for nonvisual touch screen text entry in Perkinput. In *Proc. Graphics Interface*, pages 121–129, 2012.

[2] P. O. Kristensson and S. Zhai. Relaxing stylus typing precision by geometric pattern matching. In *Proc. IUI*, pages 151–158, 2005.

[3] C. Southern, J. Clawson, B. Frey, G. Abowd, and M. Romero. An evaluation of BrailleTouch: mobile touchscreen text entry for the visually impaired. In *Proc. MobileHCI*, pages 317–326, 2012.

[4] K. Vertanen and P. O. Kristensson. A versatile dataset for text entry evaluations based on genuine mobile emails. In *Proc. MobileHCI*, pages 295–298, 2011.