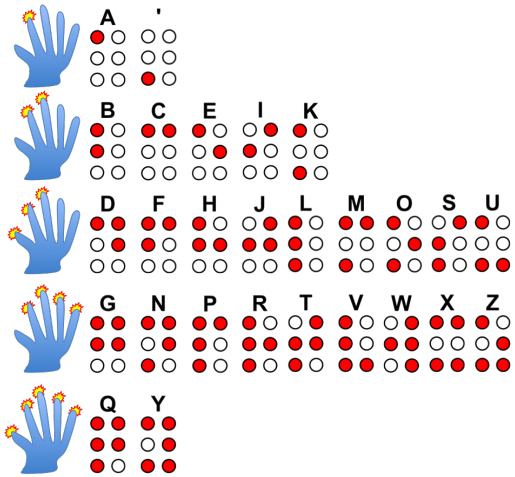
Counting Fingers: Eyes-Free Text Entry without Touch Location

Keith Vertanen
Michigan Technological University
Houghton, MI, USA
vertanen@mtu.edu



CHI'16 Workshop on Inviscid Text Entry and Beyond, 8 May 2016, San Jose, CA.

Abstract
Entering text on a touchscreen device without visual feedback can be challenging. In this paper I explore the feasibility of an approach where a single character of output is ambiguously specified by touching the screen with between one to five fingers. The count of fingers is fed to a sentence-based decoder which attempts to infer a user's intended text. This approach allows users to touch the screen with any finger orientation so long as it is touched with the correct number of fingers. I compare the accuracy of five different finger count to letter mapping, including two based on braille. Four of the mappings provided promising accuracy with character error rates below 5% on simulated input.

Author Keywords
Text entry; eyes-free text input; mobile interaction

ACM Classification Keywords
H.5.2 [Information interfaces and presentation]: Input devices and strategies

Introduction
Many computer interfaces rely on text input. Conventional desktop computers support efficient text input easily via a physical keyboard for input and a screen for visual output. The rise of mobile devices has extended the reach of computing to many new locations and situations. Increasingly

Finger count to letter maps:

- **BRAILLE** - Based on the number of braille dots. Since 5 had only two letters and q is rare, I assigned space to 5.

  1 = a', 2 = bceik,
  3 = dfhjlmosu, 4 = gnprtvwxz,
  5 = qy_

- **BRAILLE+** - Same as BRAILLE except space is indicated by some other input action.

  0 = _, 1=a', 2=bceik,
  3 = dfhjlmosu, 4 = gnprtvwxz,
  5 = qy

- **ALPHABETIC** - Alphabetic order with a similar number in each group except for space which is specified by a one finger tap.

  1 = _, 2 = abcdef,
  3 = ghijklm, 4 = nopqrst,
  5 = uvwxyz'

- **ALPHABETIC+** - Alphabetic order with an space indicated by some other input action such as a swipe.

  0 = _, 1 = abcde, 2 = fghij,
  3 = klmno, 4 = pqrstu,
  5 = vwxyz'

- **CVS** - Uses only three groups: consonants, vowels, and space.

  1 = _, 2 = aeiouy,
  3 = bcdfghjklmnpqrstvwxz'

mobile devices combine a device's input and output capabilities into a smooth glass touchscreen. But touchscreen text input is particularly difficult for users who are blind or visually-impaired. Touchscreens also present challenges for users who are situationally-impaired and operating a device eyes-free. In the future, as computing becomes even more pervasive, always-available interactions [7] may require input on devices that lack visual displays altogether.

This work explores whether simply knowing how many fingers were pressed on a touchscreen is sufficient for accurate recognition of a user's intended text. One possible application would be to provide users proficient in braille, or perhaps just learning braille, an easier way to enter text without requiring chorded input exactly specifying which braille cells are on. In grade 1 unabbreviated braille, characters are represented by a 3 x 2 binary matrix of dots. Since the letters A–Z plus apostrophe use between one and five dots, it is possible to create a direct mapping between finger count and characters according to the braille code.

Past work on chorded physical keyboards has shown experts can achieve fast entry rates (e.g. 60 wpm in [5]). But input methods such as the Twiddler keyboard require substantial practice and a specialized device. Past work has also investigated chorded input on touchscreen devices. Both Perkinput [1] and BrailleTouch [8] make use of multitouch chords specifying a braille cell. Using such multitouch chords on small devices requires either two input events using one hand or simultaneous events using both hands.

Such chorded multitouch input may be tricky to get right, both for the software, and for a user in a realistic mobile use scenario. The finger counting approach allows a user to hold the device in one hand while making a single multitouch gesture with a second hand. This frees a user to adopt any hand pose that allows all five fingers to strike the sensor. Further, a user only has to remember the number of fingers to touch with and can ignore where they touch.

Finger count to letter mappings

This finger count to letter approach is similar to an ambiguous keyboard in which a single button has multiple labels. A button press specifies a set of possible letters and a language model or dictionary determines the most likely word given a sequence of ambiguous key presses (e.g. T9).

There is a long history of work on optimizing ambiguous keyboards (e.g. [4, 2]). Here I focus on mappings that closely mirror braille or consist of easy to learn letter groupings. As shown in the list in the left margin, I designed five one-to-many mappings for the numbers 1–5 to the 28 characters: a–z, apostrophe, and space (denoted _). Numbers specify how many fingers were detected in a multitouch event. I also allow "plus" variants where space is entered by some other input action such as a right swipe (denoted by 0).

Decoder and input data

Input to the recognizer is a sequence of numbers, one number for each character in a user's desired sentence. This sequence is fed to the VelociTap decoder [10]. Normally VelociTap uses a keyboard model based on two-dimensional Gaussians. In this work I replaced the keyboard model with a step model that places all probability mass on a single key. That key is the number in the mapping and each key generates a recognition hypothesis for every character assigned to the ambiguous key. Aside from these modifications, VelociTap's search works as described in [10].

I created a 12-gram letter language model with a vocabulary of the lowercase letters a–z, limited punctuation (',!?), and space. I created a 4-gram word model with a vocabulary of the most frequent 64 K words. I trained the models

Random sentences from the evaluation set:

- maybe she will be too preoccupied about whether she has a job or not
- if you need john speak up now
- in isolation all of these things are trivial
- the troops just walked in
- it is no problem
- i'm available by cell when you get data points
- had we not run into the el paso issues the number would have held
- i hate not being there but i think these other meetings are big enough to have presence in omaha
- shelly there will be no one from my group unless i go up to meet with minnegasco
- my feeling is that once they affirmatively decide they are going to florida you should say that i rented a house and that you are going to come over and stay with us for a few days

on billions of words of Twitter, Usenet, blog, social media, forum, and movie subtitle data. Training sentences were filtered using cross-entropy difference selection [6] using an in-domain model trained on short email sentences.

For testing, I used sentences written by Enron employees on their Blackberry mobile devices [9]. I split a set of 1347 sentences containing no numbers into equal sized development and evaluation test sets. Test sentences consisted of only lowercase characters, apostrophes, and spaces. For each set, I generated ambiguous key entry sequences based on each finger count to letter mapping. This simulates a user who makes perfect use of a given mapping (i.e. the user did not mistakingly use the wrong number of fingers, add extra input, or leave out input). I used the simulated entries in offline experiments, measuring VelociTap's ability to recover the original text from the ambiguous input.

I tuned VelociTap's parameters on the development set. A different parameter set was tuned for each mapping. VelociTap normally proposes character insertion and deletions to model mistakes in touchscreen input. I disabled insertions and deletions for the experiments reported here. I used the tuned parameters to recognize the unseen evaluation set.

Results

I report recognition accuracy using character error rate (CER). CER is the number of character substitutions, insertions, and deletions required to transform the recognized text into the reference text divided by the number of character in the reference (times 100). To measure the computation costs of the different mappings, I report the average decode time per sentence on ten passes over the evaluation set. Experiments were on a i7-4790K 4 GHz desktop with four cores. VelcoiTap was set to use eight threads.

Table 1 shows the performance of the five mappings from most to least accurate. Providing a separate action for space allowed both the BRAILLE and ALPHABETIC mappings to provide much better accuracy. As might be expected, the more balanced sized groups of the ALPHABETIC mappings provided better accuracy compared with the more unbalanced BRAILLE mappings. Both BRAILLE mappings provided error rates below 4% CER. This suggests it may be possible to provide accurate recognition for braille users without requiring they learn a new mapping.

Mapping	CER (%)	Decode time (s)
ALPHABETIC+	1.96	0.14
BRAILLE+	2.77	0.11
ALPHABETIC	3.41	0.20
BRAILLE	3.69	0.21
CVS	23.17	4.31

Table 1: Error rate and average recognition time of the mappings.

The CVS mapping was a disaster. Providing just whether a letter is a vowel or consonant provided insufficient signal. This mapping did work on some sentences (21% of sentences were decoded with no errors), but for many it got the majority of words wrong. While spaces and the length of words were essentially known, VelociTap was still free to imagine many words at each position, some of which may end up being more probable under the language model than the actual sentence. Here is one example:

```
Reference   :  i like nick and everett
Recognition :  i love kids and animals
```

Impact of language model size

Thus far the VelociTap decoder has been using quite large language models (a 2.2 GB letter model and a 3.8 GB word model). This approximates the accuracy obtainable using cloud-based infrastructure. However, for performance or

Language models Combined size (MB)	ALPHABETIC+ CER (%)	BRAILLE+ CER (%)
47	3.05	4.78
260	2.34	3.60
1701	1.96	2.98
6352	1.96	2.77

Table 2: Error rate using different sized language models.

privacy reasons, decoding on device may be desirable. Table 2 shows the accuracy of the ALPHABETIC+ and BRAILLE+ mapping using different sized language models. While it appears model size can be reduced somewhat without sacrificing accuracy, getting accurate models small enough to be deployable on device may be challenging.

Future Work

Thus far I have only tested the approach using simulated perfect input. Real-world input is bound to be more imperfect: users may not always remember the right number corresponding to a character, users may not get all their fingers in the active area of the device, touchscreen sensors may detect the wrong number of fingers, etc. While the best mappings have a low error rate of 2% CER, whether they can continue to provide acceptable accuracy in the face of real-world user input noise remains to be seen.

Entry speed was not explored here. But it is likely the finger counting approach will be competitive with chorded physical or touchscreen keyboards. Experts have been shown to reach 60 wpm on a chorded physical keyboard [5] and 38 wpm on a chorded touchscreen keyboard [1]. Thus the finger counting may plausibly allow us to approach the free-flowing, or inviscid entry rate required to compose novel text. This rate has been estimated at around 67 wpm [3].

Another open question is whether users can easily learn such a mapping and actuate the required multitouch gestures. Finally, it would be interesting to explore whether the braille mappings are useful either as an input method itself or as a training aid for users who are visually-impaired.

Conclusions

This preliminary work shows that using only the count of fingers in a touchscreen event may provide sufficient signal for accurate sentence-based recognition. Adopting a mapping based on the number of dots in a braille cell allowed sentences to be input with a low error rate of 3.7% CER. If an additional gesture can be used for space, error rate dropped further to 2.8% The finger counting approach has the potential to be an easier input method to learn than other braille-based input methods that require the explicit specification of the entire braille cell pattern.

A mapping based on alphabetic ordering provided even better accuracy than the braille mappings with a CER of 3.4%. An even lower CER of 2.0% was possible using an extra gesture for space. The alphabetic mappings could provide an easy-to-learn input method for devices without a visual display or use in situations where users cannot visually attend to their device.

Author biography

Keith Vertanen is an Assistant Professor at Michigan Technological University. He specializes in designing intelligent interactive systems that leverage uncertain input technologies including input via speech, touch, and eye-gaze.

Workshop demo

For demonstration at the workshop, I plan to implement several of the mappings in a standalone Android text entry app using language models that fit on typical devices.

References

[1] Shiri Azenkot, Jacob O. Wobbrock, Sanjana Prasain, and Richard E. Ladner. 2012. Input Finger Detection for Nonvisual Touch Screen Text Entry in Perkinput. In *Proceedings of Graphics Interface 2012 (GI '12)*. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 121–129. http://dl.acm.org/citation.cfm?id=2305276.2305297

[2] Mark D. Dunlop, Naveen Durga, Sunil Motaparti, Prima Dona, and Varun Medapuram. 2012. QWERTH: An Optimized Semi-ambiguous Keyboard Design. In *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services Companion (MobileHCI '12)*. ACM, New York, NY, USA, 23–28. DOI: http://dx.doi.org/10.1145/2371664.2371671

[3] Per Ola Kristensson and Keith Vertanen. 2014. The Inviscid Text Entry Rate and Its Application As a Grand Goal for Mobile Text Entry. In *Proceedings of the 16th International Conference on Human-computer Interaction with Mobile Devices and Services (MobileHCI '14)*. ACM, New York, NY, USA, 335–338. DOI: http://dx.doi.org/10.1145/2628363.2628405

[4] G.W. Lesher, B.J. Moulton, and D.J. Higginbotham. 1998. Optimal character arrangements for ambiguous keyboards. *Rehabilitation Engineering, IEEE Transactions on* 6, 4 (Dec 1998), 415–423. DOI: http://dx.doi.org/10.1109/86.736156

[5] Kent Lyons, Thad Starner, Daniel Plaisted, James Fusia, Amanda Lyons, Aaron Drew, and E. W. Looney. 2004. Twiddler Typing: One-handed Chording Text Entry for Mobile Phones. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. ACM, New York, NY, USA, 671–678. DOI: http://dx.doi.org/10.1145/985692.985777

[6] Robert C. Moore and William Lewis. 2010. Intelligent Selection of Language Model Training Data. In *Proceedings of the ACL 2010 Conference Short Papers (ACLShort '10)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 220–224. http://dl.acm.org/citation.cfm?id=1858842.1858883

[7] Dan Morris, T. Scott Saponas, and Desney Tan. 2011. Emerging Input Technologies for Always-Available Mobile Interaction. *Foundations and Trends in Human-Computer Interaction* 4, 4 (April 2011), 245–316. DOI: http://dx.doi.org/10.1561/1100000023

[8] 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 (MobileHCI '12)*. ACM, New York, NY, USA, 317–326. DOI: http://dx.doi.org/10.1145/2371574.2371623

[9] Keith Vertanen and Per Ola Kristensson. 2011. A Versatile Dataset for Text Entry Evaluations Based on Genuine Mobile Emails. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI '11)*. ACM, New York, NY, USA, 295–298. DOI: http://dx.doi.org/10.1145/2037373.2037418

[10] 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 (CHI '15)*. ACM, New York, NY, USA, 659–668. DOI: http://dx.doi.org/10.1145/2702123.2702135