SHRIMP - Solving Collision and Out of Vocabulary Problems in Mobile Predictive Input with Motion Gesture

Jingtao Wang ♦ **Shumin Zhai** § **John Canny** ♦

♦Computer Science Division
UC Berkeley, 387 Soda Hall, Berkeley, CA, USA
{jingtaow, jfc}@cs.berkeley.edu

§IBM Almaden Research Center
650 Harry Road, San Jose, CA, USA
zhai@almaden.ibm.com

ABSTRACT

Dictionary-based disambiguation (DBD) is a very popular solution for text entry on mobile phone keypads but suffers from two problems: 1. the resolution of encoding collision (two or more words sharing the same numeric key sequence) and 2. entering out-of-vocabulary (OOV) words. In this paper, we present SHRIMP, a system and method that addresses these two problems by integrating DBD with camera based motion sensing that enables the user to express preference through a tilting or movement gesture. SHRIMP (Small Handheld Rapid Input with Motion and Prediction) runs on camera phones equipped with a standard 12-key keypad. SHRIMP maintains the speed advantage of DBD driven predictive text input while enabling the user to overcome DBD collision and OOV problems seamlessly without even a mode switch. An initial empirical study demonstrates that SHRIMP can be learned very quickly, performed immediately faster than MultiTap and handled OOV words more efficiently than DBD.

Author Keywords

Text Input, Mobile Devices, Predictive Input, Dictionary-Based Disambiguation, Gestures, Mobile Phones, Camera Phones, MultiTap, T9.

ACM Classification Keywords

H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces; Input devices and strategies, Theory and methods.

General Terms

Design, Experimentation, Human Factors.

INTRODUCTION

With an estimated 4 billion units in use in December 2008 [9], mobile phones have already become the most popular computing device in human history. Their portability and communication capabilities have revolutionized how people interact with each other. However, despite the rapid growth of mobile phones, text entry on small devices remains a major challenge. Due to trade-offs in both size and compatibility, most mobile phones today are equipped with a 12-key keypad (Figure 1). This keypad is effective for dialing phone numbers but not for editing contact lists, composing SMS messages or writing emails. Other input devices, such as mini QWERTY keyboards and touch screens are on the rise, but the 12-button keypad-based mobile phones are still, and will likely to be for years to come, the majority in the market.

One fundamental difficulty in text entry using a 12-key keypad is that the mapping of 26 alphabet characters to the 12 keys is inherently ambiguous. In the ITU E.161 standard [8], one numeric button corresponds to 3 or 4 alphabet characters on the keypad (Figure 1). All mobile text input techniques relying on the 12-key keypad have to resolve the ambiguity that arises from this one-to-many mapping.

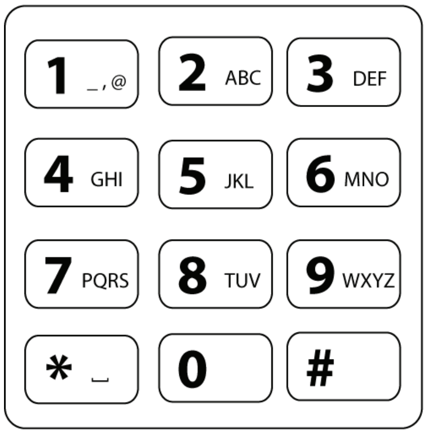

Figure 1. The standard 12-key telephone keypad, character layout follows the ITU E.161 standard [8]

Most disambiguation methods can be categorized into the following two categories:

Action Based Disambiguation. Users rely on multiple key presses (e.g. MultiTap, TNT [7]), concurrent chording [19], tilting [20] or motion [21] to select one character from the multiple alphabetical characters on each key.

Linguistic Disambiguation. Also known as predictive input, these methods use redundant information in language to disambiguate users' input when entering standard English words. Linguistic knowledge can be leveraged either through Dictionary-based Disambiguation (DBD e.g. T9 [5]), character level N-gram models (e.g. LetterWise [14]), or a combination of both. Besides disambiguating uncertain input strings, linguistic knowledge can also be used for predicting users' future intention (a.k.a. word completion [24, 25]).

Methods in both categories have their unique strengths and weaknesses. Action based disambiguation allows users to enter any character deterministically, but requires additional sequential or concurrent actions. DBD input techniques such as T9 can achieve approximately 1.007 KSPC (Key Stroke Per Character) for *words that are in the dictionary* [14]. However, they depend on an alternative input method to enter words that are not in the dictionary known as out-of-vocabulary (OOV) words and suffer from the encoding collision problem (to be detailed in the next section). Character level n-gram model based disambiguation, such as LetterWise [14], can achieve a KSPC that is close to DBD and works for OOV words; however, continuous visual attention is required to confirm suggested characters after each key press.

In this paper, we present a novel method called SHRIMP[1] (Small Handheld Rapid Input with Motion and Prediction) which enable the user to handle the collision and OOV problems of DBD input more easily. SHRIMP is a predictive text input method based on Vision TiltText [21] and runs on camera phones equipped with a standard 12-button keypad. SHRIMP is as effective as conventional DBD when entering unambiguous in-dictionary words. SHRIMP uses seamlessly integrated concurrent motion gestures to handle ambiguous dictionary words or OOV words without mode switching.

RELATED WORK

MultiTap

MultiTap is perhaps the most popular text entry method for mobile phones. It requires the user to press the key labeled with the desired character repeatedly until the correct character appears on the screen. MultiTap is simple, unambiguous but tedious. It has an average KSPC of 2.03 [17].

Vision TiltText

Vision TiltText by Wang *et al* [21] is a remake of the TiltText input method by Wigdor and Balakrishnan [20]. Instead of using an accelerometer, Vision TiltText uses the built-in camera on a phone to detect motion so that it can run on unmodified mainstream camera phones. Implementation details can be found in [20] and [21]. With the help of a concurrent gesture movement, Vision TiltText can achieve 1 KSPC on any character. However, moving a cell phone left and right for entering about 60% of characters is an overhead which can be further reduced. In this paper, we present SHRIMP that combines Vision TiltText with DBD. SHRIMP can minimize the concurrent movement requirement of Vision TiltText when entering in-dictionary words but leverage vision TiltText when entering OOV words.

Predictive Input

Dictionary based disambiguation (DBD) has been well-researched since the 1970s [18]. A popular commercial implementation of this kind is marketed as T9 by Tegic Communications, a former subsidiary of AOL and now Nuance Communications Inc. DBD uses a dictionary to detect all the possible words that match users' numeric keypad input. For example, the numeric sequence `2-6-6-7-8-8-3-7` will result in "`computer`" because that is the only English word in the dictionary that meets the constraints defined by the input string, When multiple words in the dictionary map to the same numeric string (encoding collision), manual selection is needed if the intended word is not displayed as the first choice. For example, the sequence `6-3` may mean either "`of`" or "`me`", while the sequence `2-5-6-8-3` could mean "`cloud`", "`aloud`" or "`clove`". In an extreme situation, this encoding collision problem has caused the "SMS generation" to accept "`book`" as "`cool`" since the former is more frequent in formal English hence presented as the first choice and many users don't bother to change it to the latter [22]. Language models [12] can be used to predict the more likely one from the colliding candidates [16], but they cannot totally eliminate uncertainty. Another problem is that DBD only works when the user enters English words that are in the dictionary. Both Dunlop and Crossan [2] and Mackenzie *et al.* [14] questioned the flexibility of such a dictionary based approach. Person names, place names, company names, new product names, abbreviations, acronyms, or combinations of letters and numbers are frequently used in mobile environment but are not likely in the dictionary. According to Jansen *et al.* [11], 20% of Twitter posts mention specific product and brand names. Dunlop summarized the dilemma well: "*the dictionary method is not a sufficient input method on its own but having two modes is likely to lead to great confusion*" [2].

OOV words can be handled with additional time consuming work around steps. For example, the following approach is usually implemented in commercial products running DBD input such as T9 (used in mobile phones from Nokia,

---

[1] The method presented in this paper, SHRIMP, is named in the tradition of SHARK [27] and Fisch [24]. SHARK requires a relatively large touch screen for the virtual keyboard overlay; Fisch is originally designed for a small touch screen and can be extended to a touch-ball or a joystick [25], SHRIMP works on unmodified camera phones equipped with a standard 12-key keypad.

Samsung LG and others). For an OOV word, the user can use the "UP" and "DOWN" arrow key to navigate though the character candidates from each key press and use the "RIGHT" arrow button to confirm character after character. Similarly, collision words are handled by navigating through the multiple matching words via the "UP" and "DOWN" arrow if the intended word is not the most frequent one. In this paper, we show that SHRIMP is a dictionary based predictive input method that supports the entry of any word *efficiently* without mode switching.

THE DESIGN OF SHRIMP

DBD as Regular Expression Matching

Predictive text entry via DBD can be considered a regular expression [1] matching problem and SHRIMP can be defined as a natural upgrade of DBD under the regular expression matching framework.

When a user starts to press button '2' in DBD, he/she tells the system that the intended word starts with either 'a', 'b', or 'c'. This kind of constraint can be captured by the regular expression /^[abc]$/[2] ; if the user follows with a key press '6', the regular expression is extended as /^[abc][mno]$/. If the user finishes the whole word with two more characters '6' and '5', the regular expression becomes /^[abc][mno][mno][jkl]$/. As a result, DBD input can be considered as a query to the dictionary to retrieve all words that match the given regular expression. In this example, the words "book" and "cool" are the candidates that match the regular expression and would be the output of DBD.

SHRIMP as an Extension of DBD

Pressing a numeric button in DBD can be considered as adding a [wxyz] style constraint to the existing regular expression. However, it is not the only constraint we can add from a regular expression perspective. If we use a Vision TiltText gesture to enter character 'c' at the beginning, i.e. press and hold button '2', move right, release. This action will tell the computer that the intended word starts with 'c' rather than [abc], the corresponding regular expression for this action will be /^c$/. If we continue the input without motion by typing 6-6-5, the final regular expression will become /^c[mno][mno][jkl]$/ and 'cool' will be the only word that matches the given regular expression. Vision TiltText can be incorporated at any stage of the word entry. Any time a motion gesture is used, it is telling the system that only one character, determined by Vistion TiltText, should appear at the current character location and the specific character, rather than the bracket enclosed character set will be used to construct the corresponding component in the regular expression. Figure 2 shows the steps and corresponding screen output to enter word 'cool' by using MultiTap, DBD and SHRIMP (here motion based constraint is used to enter the last character 'l').

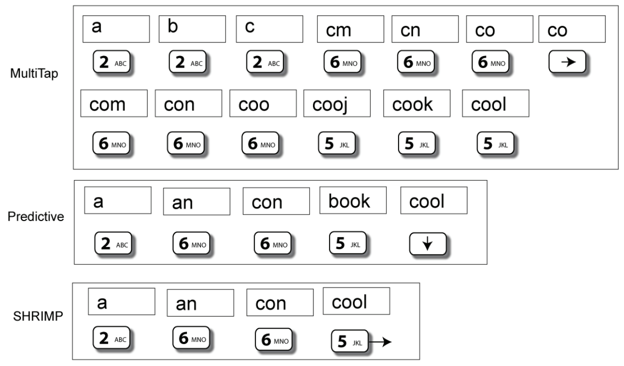

Figure 2. Using MultiTap, Predictive Input (DBD) and SHRIMP to enter a word 'cool' with encoding collision

DBD and Vision TiltText can be considered two extreme cases of SHRIMP – if we type every character without motion, then the output of SHRIMP will be no different from DBD. If we type each character with motion constraints, then SHRIMP becomes Vision TiltText so OOV words can be entered easily. Figure 3 shows how DBD and SHRIMP could be used to enter an OOV word – "hci". SHRIMP can save four out of the seven key strokes required by DBD with concurrent motion. Note that the character level confirmation approach that appeared in commercial products is used in DBD to enter OOV word in this example. If the user chooses to switch to MultiTap, the total number of key presses would become – 1(switch to MultiTap) + 2 (h) + 3(c) + 3(i) + 1(switch back to DBD) = 10.

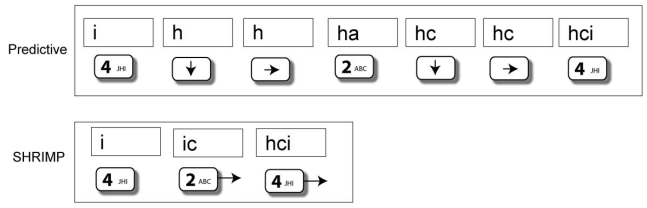

Figure 3. Using DBD and SHRIMP to enter out of the dictionary words

There is one more case that needs additional consideration in SHRIMP – the action of typing a key without movement. It could mean the user did not bother to express his/her preference so he/she may intend any of the characters shown on that key. Alternatively, it could mean the user intended to type the middle character on the button via Vision TiltText (by default, no motion means entering the middle character in Vision TiltText). Such ambiguity can be addressed with two different approaches. First, we can

[2] Symbol ^ and $ in the regular expression marks the beginning and end of a character string. [abc] means that either a, b or c could be a valid match.

redefine typing the middle character in Vision TiltText as a "press"-"move up"-"release" action to eliminate such kind of ambiguity completely. Second, the following heuristics can be used - we first assume the no movement situation as DBD style ambiguous input to construct the regular expression. If some words in the dictionary can match the given regular expression, and the Vision TiltText style unambiguous interpretation corresponds to an OOV word, then the OOV word will be added as the last candidate just in case the user intends to type the OOV word rather than the word within the dictionary. If no word matches the given regular expression, then we can interpret *every* no-movement key press as typing the middle character via Vision TiltText. We have implemented both approaches in our prototype and have found that the second approach more convenient in actual usage. This is because when entering an OOV word, the users have to type every character in Vision TiltText and the chance that this constrained input will match existing words in the dictionary is small (please refer to the next section for details).

SHRIMP has three major advantages when compared with other linguistic based disambiguation or word completion methods. First, SHRIMP can enter both in-dictionary words and OOV words, without mode switching. Second, searching for correct candidates visually after each key press is not required; users only need to visually confirm the result at the end of a word. Third, SHRIMP provides a smooth learning curve for beginners – it is not necessary to remember when and where to add motion constraints and adding more motion constraints than necessary won't change the input result (as illustrated in the next section – adding one, at most two motion constraints will disambiguate encoding collisions in the worst case). Even if a user completely ignores to use motion constraints, SHRIMP is equivalent to DBD in this worst case. Different from DBD, SHRIMP provides users an opportunity to enter "troublesome" words more effectively next time. If the same problematic word shows up frequently (e.g. "`book`" for "`cool`"), a user is more likely to use the motion based gesture to deal with it next time. In short, SHRIMP allows the user to be more "expressive" than traditional DBD, but to a completely voluntary degree. There is no downgrade from DBD for not using the additional expressive power through motion gesture.

In the next section, we show the feasibility and power of SHRIMP via corpus analysis.

ANALYSIS

Analyzing text entry performance realistically is a challenging task. For the current topic, we need to consider three different types of words: 1. words in the dictionary without encoding collision, 2. words in the dictionary with encoding collision, 3. OOV Words. Previous studies tend to focus on words in categories 1 and 2 [2].

As discussed in the previous section, SHRIMP is similar to DBD when typing words in the dictionary without encoding collisions (Type 1 words). When typing words with encoding collisions (Type 2 words), some of the characters can be entered with motion sensing to reduce ambiguity. When typing OOV words (Type 3 words), users need to provide a motion gesture for each character, and the experience of SHRIMP is no different from Vision TiltText. As a result, the actual performance of SHRIMP vs. DBD will depend on the distribution of the three types of words in the user's writing.

In order to understand the distributions of these types of words, we performed quantitative analyses on two text corpora. The first is the Brown Corpus [13] from which we extracted the top 17805 words based on frequency rank, stripped punctuations and normalized them to lower case. This corpus is similar to the one used in [24]. We used this corpus to study the encoding collision problem in DBD. The second corpus is the NUS SMS Corpus [6]; it has 7189 distinct words representing text messaging vocabulary. This corpus gives an opportunity to study the OOV problem in mobile environment.

Two types of encoding collision analysis were conducted - raw (treating each distinct word with the same weight) and frequency weighted (each word weighted by its frequency of occurrence calculated from the corpus). The raw analysis gives a sense of proportion to all unique words including rare words. The weighted frequency analysis is proportional to natural occurrence in real use (according to the word frequency distribution in the corpus used).

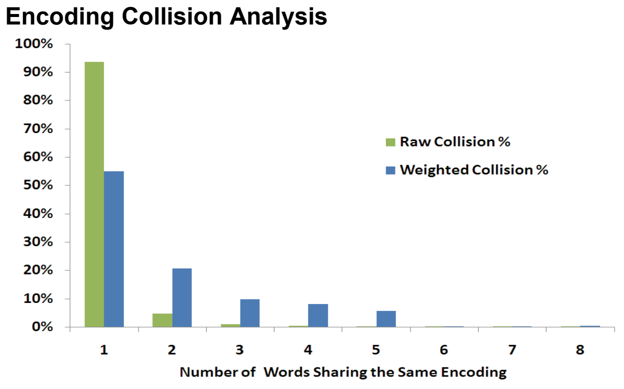

Figure 4. Word distribution in different collision categories according to the number of words sharing the same key code

Figure 4 shows the distribution of words encoding collisions on the standard 12-key keypad (Figure 1) in the Brown Corpus. The horizontal axis is the category of collisions (namely the number of words sharing the same key press sequence). For example, the percentage shown in the first two bars (raw and frequency weighted percentage respectively) where the collision number is 1 are the percentage of words without encoding collision. From Figure 4 we see that there can be as many as eight words that collide on the same numeric encoding. For DBD, 6.4% of words have encoding collisions in the raw analysis. In the weighted analysis, the collision rate raises to 44.9%. Not all of these 45% collisions would require additional action of the DBD users. For example, when two (and only two) words collide on the same code, outputting the word with higher frequency will be correct in more than 50% of the time by definition. More complicated prediction methods based on for example a word level N-gram statistical model may further increase the chance of success when multiple words are in collision. However, not only these methods may require the amount of CPU and memory still not available on mobile phones, statistical models based on large and formal corpus analyses may not apply well to actual mobile use at all. The previously mentioned `book` vs. `cool` is one example.

Which buttons do words tend to collide on? The analysis summarized in Figure 5[3] shows that based on the weighted analysis most ambiguities appear on button "3", "6", "4", "8", corresponding to the character groups `[def]`, `[mno]` `[ghi]` and `[tuv]`. So these buttons can benefit the most from SHRIMP's motion gesture enhancement. In contrast, words rarely collide on buttons `5[jkl]` and `9[wxyz]`. A possible design implication is to graphically mark the collision prone buttons in a way that encourages the use of motion gesture on them.

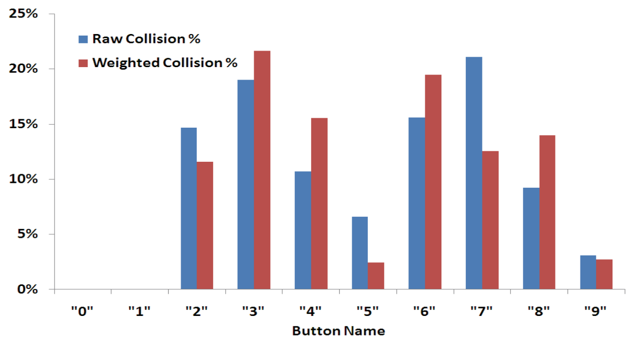

Figure 5. Collision distribution by Numeric Button

As previously stated, if motion gesture is used for every button press, then SHRIMP becomes Vision Tilt Text and the output will be completely unambiguous. If motion gesture is only used on some of the button presses, then the frequency of word collisions can still be reduced although not completely eliminated. Figure 6 and 7 show the result of using hypothetical strategies of partial motion gesture use: on first letter only, on last letter only, on both first and last letter, on the first two letters, and on the last two letters of a word. Compare them with Figure 4 one can see the dramatic reduction in collision with these partial use strategies. For example if we only use motion gesture to disambiguate the first key press, the encoding collision will drop from 6.4% in DBD to 3.1% in the raw analysis (a 52% drop) and from 40.4% to 21.9% in the weighted analysis.

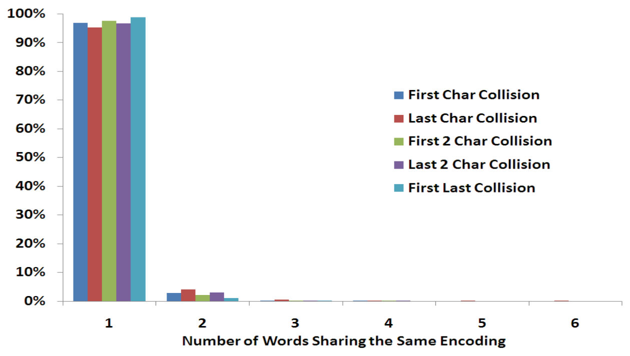

Figure 6. Raw SHRIMP Word Collision Frequency by Typing Strategy and Number of Words Sharing the Same Encoding

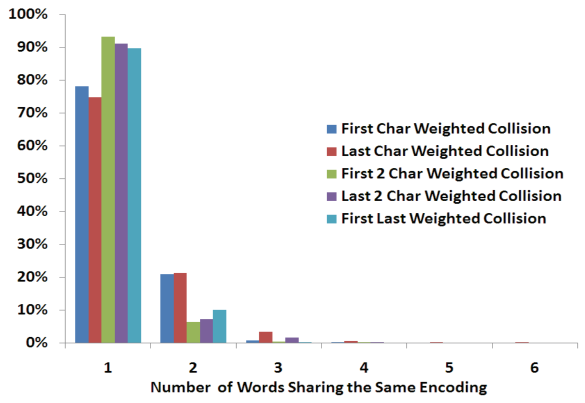

Figure 7. Weighted SHRIMP Word Collision Frequency by Typing Strategy and Number of Words Sharing the Same Encoding

When motion gesture is used with SHRIMP, even only some of the times (e.g. on first and last only), the maximum number of words colliding when collision do occur can be reduced to less than three. Three words can be disambiguated easily selected through a space button press coupled with a motion gesture (left, none, right).

OOV Analysis

It is well known that the users' vocabulary in mobile text entry is different from formal written English collected in more formal text corpora [2, 14]. However, due to the lack of large scale, publically available text corpora from mobile users, it is difficult to quantify the difference between formal written English and the language people use in mobile text entry. To our knowledge, the only publically available mobile text corpus is the NUS SMS Corpus [6]. Despite its relatively small size (121k words with 7189 distinct words), it has been adopted by HCI researchers to simulate word frequencies in mobile applications [4]. The corpus is a collection of self-submitted SMS messages from students at National University of Singapore (NUS). Because of the limited sample size, it is difficult to estimate

[3] Button "0" and button "1" are not used for encoding alphabet characters in the ITU E.161 standard [8].

how representative this corpus is when compared with the whole mobile population. As a result, the NUS SMS corpus could only serve as a case study on the OOV problem.

Based on our analysis, the NUS SMS word frequency is quite different from that in traditional written English. For example, the top 8 words in the Brown Corpus - "`the, of, and, to, a, in, that, is`" have little overlap with the top 8 words in the NUS SMS Corpus - "`i, u, to, me, at, my, go, you`". Only the word "`to`" appears in both lists. The top three words in formal English – "`the, of, and`" did not even make it into the top 8 list of NUS SMS corpus.

Our analysis shows that only 49% of the words in the SMS Corpus can be found in the traditional word corpus by raw frequency. In the frequency weighted calculation, only 36% of the words in SMS can be found in the traditional word corpus. Sample out-of-the-dictionary words include "`lor, liao, okie, slackin, thkin`". This means that if we create the DBD dictionary by parsing a traditional corpus based on written English and use a DBD method based on that dictionary to enter the NUS SMS Corpus, 64% of the intended entries will be OOV.

As mentioned earlier, a fully 20% of Twitter micro blogs contain product and brand names [11]. A very high percentage of these names are likely to be OOV. Twitter imposes a 140 character limit per posting. This limit, plus the relatively slow process of entering text on mobile phones in general, are likely to encourage users to create ad hoc abbreviations that are OOV.

In conclusion, although we have only one quantitative case study available, it is safe to say a large percentage of words in SMS and Twitter like applications are OOV. The problems of collision and OOV with the traditional DBD method are frequent enough to warrant a SHRIMP or SHRIMP like solution.

A follow up question is when a user enters an OOV word, would he/she realize the word is OOV therefore engages SHRIMP's motion gesture to preempt DBD's failure or the need to switch to a different mode? The question is important to both DBD and SHRIMP and will be addressed in the user study reported later in this paper.

IMPLEMENTATION

Although lacking in some mobile HCI research, it is important to test mobile interface concepts and methods with real mobile device for both engineering feasibility and interaction form factor reasons. We have made a complete implementation of SHRIMP on a Motorola v710 phone (a CDMA Phone from Verizon Wireless). This was a common off-the-shelf camera phone at the time of our implementation. The v710 has an ARM9 processor, 4M RAM and a 176x220 pixel color display. Our application is written in C++ in BREW 2.11 (the Binary Runtime Environment for Wireless, http://brew.qualcomm.com). We used the Realview ARM Compiler 1.2 for BREW to cross-compile the target application. We used the TinyMotion library [21] for real time camera phone based motion sensing. The compiled target application, including the implementation of MultiTap, DBD, Vision TiltText, SHRIMP, as well as a 15k words dictionary and the corresponding index data is around 1.3MB in size[4] . The required runtime memory is about 250k. At this time, we have also ported SHRIMP to a Motorola RAZR V3 cell phone that runs BREW 3.12. We believe that porting our code to other platforms, such as Windows Mobile, Symbian and Android, would be straightforward.

DBD

The programming of DBD is straightforward. Particularly worth noting is how collision and OOV are handled in its UI. When there is a word collision, our implementation of DBD allows the user to use the "UP" and "DOWN" arrow buttons to navigate through the *word* candidate list and use the space key (i.e. the star button in Motorola V710[5]) or the "RIGHT" arrow button to confirm. Basic DBD input methods [10] cannot handle OOV words. In our implementation, in addition to allowing the user to switch to another input method such as MultiTap, we also implemented an extension to DBD that is available in most Nokia, Samsung and LG phones. When entering an OOV word, the "UP" and "DOWN" arrow button can be used to navigate though the candidates of each *character* and use the "RIGHT" arrow button to confirm character after character. Most users in our user study prefer this extension rather than switching to and from MultiTap to enter OOV words.

SHRIMP

We implemented SHRIMP as a natural extension of both Vision TiltText and DBD, so most of the operation conventions in Vision TiltText and DBD are kept intact. For instance, users can press button '`2`', hold it, move/tilt the phone to the left until a vibration is felt and then release the button to indicate that character '`a`' is desired. The user can also use 'UP' and 'DOWN' buttons like in DBD to correct an ambiguous word. Of course, the candidate list is much smaller than that of DBD if additional motion constraints are used. The same 70 ms vibrato-tactile feedback mechanism [21] is also used in SHRIMP to signal that the critical movement amount has been reached.

AN INITIAL EMPIRICAL STUDY

The quantitative analyses presented have shown the following. 1. Collision and OOV are both quite frequent

[4] In our implementation, DBD and SHRIMP shares the same dictionary and a major portion of the index data, the input method code plus the motion sensing code alone, is about 150k in size.

[5] The space button could be assigned to the pound ('`#`') key or the zero ('`0`') key in other cell phones.

problems for DBD; 2. SHRIMP can significantly reduce collision even if motion gestures are only used sparingly (on first letter or only when a word has been known to be troublesome from previous experience); 3. SHRIMP can handle OOV more effectively than previous DBD workaround methods. To complement these analytical findings, we also conducted an empirical study to do an initial but holistic test of SHRIMP as an everyday text entry method. We had two basic goals for the study. One was to figure out whether or not the idea behind SHRIMP is easy to understand and if the current SHRIMP implementation is easy to learn. The second goal was to evaluate the initial performance of SHRIMP in comparison with existing text entry methods such as MultiTap, DBD and Vision TiltText. A successful mobile text entry method should not have a steep learning curve. The users should be able to pick up the method in a few minutes, and users should gain immediate benefits. Otherwise, many users may give up and switch back to older methods they were accustomed to. As part of the study we also measured users' ability to recognize OOV words. Analyzing the longitudinal performance of SHRIMP, which may reveal its advantage in handling collision, and a systematic empirical comparison across different word categories (unambiguous, collision, and OOV words) are beyond the scope of this paper and will be deferred to future work.

Study Design

Our study consisted of five parts.

Overview. We gave a brief overview of the tests to be conducted and demonstrated the four text entry methods, MultiTap, DBD, Vision TiltText and SHRIMP, to the participants. To help them better understand the motion sensing idea behind Vision TiltText and SHRIMP, we also gave a brief introduction of TinyMotion and demonstrated other motion sensing applications to the user. We let the users play with the demo applications and answered their questions. This session lasted 10 to 15 minutes.

Pre-test Questionnaire. In this session we collected basic information of the study participants including gender, educational background, current occupation, experiences with cell phones and camera phones, frequency of mobile tasks such as voice communication, SMS usage, picture taking, use of other mobile phone applications etc.

Out of Vocabulary Word Recognition. In this session we tested participants' ability in identifying OOV words. We told users that, similar to a spell checker, many mobile text entry methods maintain a list of words to predict users' intended words based on the keypad presses. We told the users that the input methods to be tested used a word list of 15K of the most popular words and showed them samples of words in the list as well as words not in the word list. We then gave the participants a list of 21 words and let them identify each of them as in-dictionary or OOV. It's worth noting that none of the OOV words in this section appeared in the follow-up text entry tasks and the subjects didn't know their performance on the OOV recognition task throughout the study. So the confounding effect between the OOV recognition task and the actual text entry task was minimal.

Text Input. In this session, we compared the performance of four mobile text entry methods – MultiTap, Vision TiltText, DBD and SHRIMP. The testing phrases were selected randomly from MacKenzie's text entry test phrase set. The timeout for the MultiTap method was 2 seconds. DBD and SHRIMP share the same dictionary, which has 15k words sorted by word frequency. Due to time constraints, each participant entered 12 sentences with a total of 52 words for each input method. Although a popular source of testing phrases used in recent studies on text entry such as [20], MacKenzie's phrase set has a limitation to our study – most of the sentences were formal and "correct" English and the word frequency in this phrase set were designed to simulate the corpus of formal written English [14, 15], not those used in SMS or other mobile applications. However, we felt these phrases would still serve the purpose of this initial pilot study – to test if users can understand and use SHRIMP without much practice. A Motorola V710 mobile phone loaded with our application was used in the experiment.

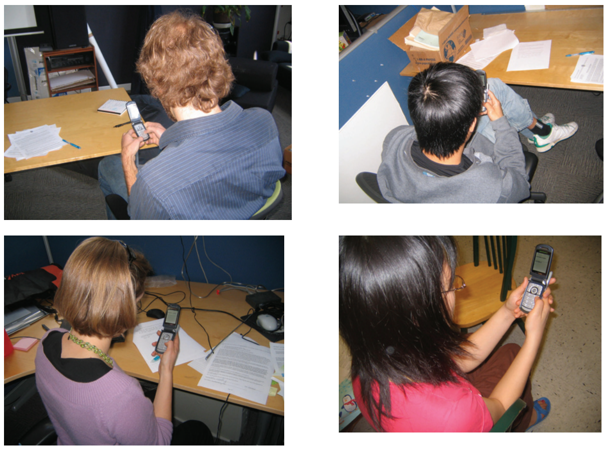

Figure 8. Sample pictures taken from the user study

This session started with a warm up practice phase in which the users could practice with the four methods tested for as long as they desired. Most of them choose to practice for 2 to 10 minutes before the actual test. The order of the four input methods was balanced via the order four Latin square patterns.

Collecting qualitative feedback. We conducted a final survey immediately after a participant completed all the sessions. In the survey the participant completed a questionnaire and commented on the input methods they tested.

To simulate the real world situations of cell phone usage, we did not control the environment used for the study. The participants were encouraged to choose their desired locations to complete the study. Most of the studies were completed in the participants' own chair or at a public discussion area in a lab. Figure 8 shows some of the actual environments used during the study.

All our input methods run on real, unmodified cell phone hardware and confounding factors such as dictionary size, user interface and screen resolution have been controlled. We will release the source code of all four text entry methods, data collection application and log processing application under the BSD license. We hope it can establish a standard and realistic testing platform for mobile text entry research and provide a baseline for future studies.

Participants

Twelve people participated in the study. Nine of them were undergraduate or graduate students in a university, two were visiting researchers of the university and one was an instructor at a local high school. Three of the participants were female and nine were male. All of them owned a cell phone at the time of the study and 11 of the 12 cell phones were camera phones. On average they had 7.5 years of experience in using cell phones (stddev = 2.2). 10 out of 12 cell phones were equipped with the standard 12-key keypad. One cell phone had a mini-QWERTY keyboard (Palm Centro) and one cell phone was touch screen only (Apple iPhone). 9 participants reported that MultiTap was their primary text entry method on cell phones; The other three participants used DBD, mini-QWERITY keyboard, or a virtual on-screen keyboard for mobile text entry. All of the participants completed all the five sessions in our user study.

EMPIRICAL RESULTS

Out of Vocabulary Word Recognition

Participants correctly identified OOV words 97.6% of the time. No in-dictionary word in the test was incorrectly recognized as OOV. Among the 6 OOV recognition misses, the person name "`Carlson`" was incorrectly categorized as "in the dictionary" four times. Similarly, "`Yvette`" was incorrectly categorized as "in the dictionary" two times. Both participants who mislabeled "`Yvette`" also mislabeled "`Carlson`". From this preliminary test, it seemed that people were overall fairly proficient at estimating whether a word was in the vocabulary even if they only had a few samples from the dictionary. In our experiment, the participants had no problem in identifying OOV business names, place names, and abbreviations. It was more difficult estimating the popularity of person names. People's social networks may have a huge impact on their perception of "frequent names".

Text Input

In total 14281 characters were entered (including white space and editing characters). There were 41 unique words in the test sentences. 17 of them had no encoding collisions and 21 of them had encoding collisions. Among the 21 words that had encoding collisions, 16 of them do not require explicit action of the user if we output the words with the highest word frequency from the candidate list. 2 words were OOV words.

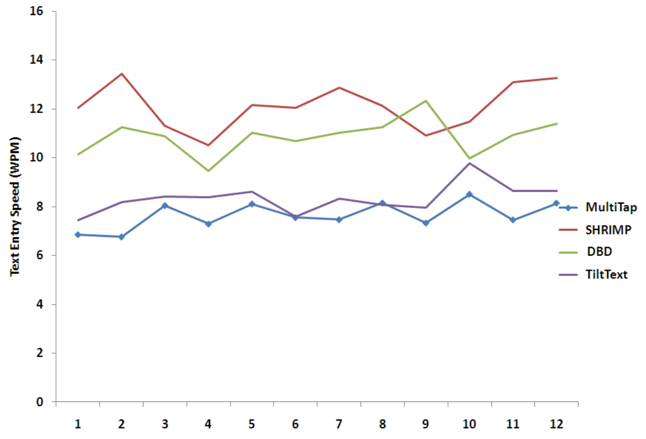

Figure 9. Text entry speed (WPM) by technique and sentence number for the entire experiment

Figure 9 shows the speed of the four different text entry methods we have tests in the pilot user study. As stated in the experimental design section, users started the tests only after 2 to 10 minutes of practicing. All of the users had previously used MultiTap (with an average of 5.3 years of experience) while only one user had used DBD frequently before our study. So the results on Vision TiltText, DBD and SHRIMP can be viewed as users' initial text entry speed without much practicing. A longitudinal study is needed to understand the expert performance of these methods. From Figure 9, we can see that SHRIMP and DBD are already faster than MutiTap and Vision TiltText when typing the first three sentences.

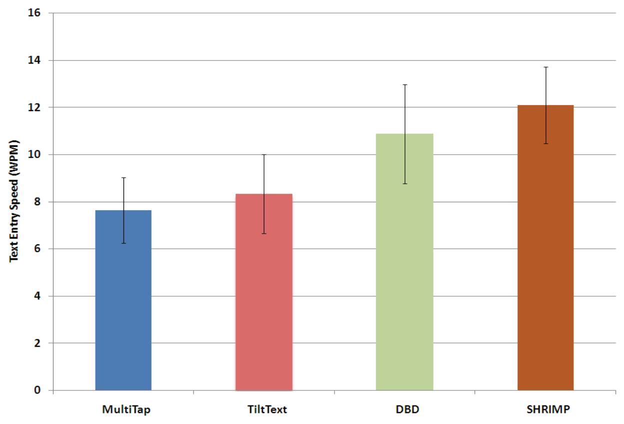

Figure 10. Text entry speed from the experiment

As shown in Figure 10, input speed varied from one method to another. Repeated measure variance analysis showed significant difference due to input method: $F(3, 33) = 110.9$, $p < .0001$. Fisher's post hoc tests showed that the speed of SHRIMP was significantly higher than the speed of MultiTap ($p < 0.001$) or the speed of Vision TiltText. The average speed of SHRIMP (12.1 wpm) was higher than that of DBD (10.86 wpm), the difference was significant in paired-sample t-Test (p < 0.001) but not significant in two sample t-Test (p = 0.12). DBD was significantly faster than MultiTap (p < 0.001). Vision TiltText (8.34 wpm) was faster than MultiTap (7.64 wpm) in average, but the difference was not statistically significant (p = 0.07).

As we discussed earlier, the testing phrases in the study resembled formal English. As a result there were only two OOV words in the 12 sentences. The close performance of DBD and SHRIMP in Figure 10 was in part caused by the relative low percentage of OOV words in our test sentences. As a follow up analysis, we isolated logs for OOV words. As shown in Figure 11, the text entry speed of DBD (3.3 wpm) dropped drastically when entering OOV words. In fact DBD became the slowest input method among all four methods. The speed of DBD for OOV words was significantly lower than MultiTap (p < 0.001), Vision TiltText (p < 0.001) and SHRIMP (p < 0.001). The speeds of SHRIMP (9.3 wpm) and Vision TiltText (9.5 wpm) in handling OOV were not significant (p = 0.059).

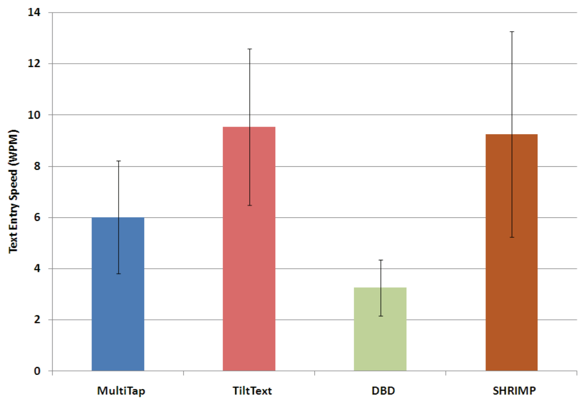

Figure 11. Text entry speed for the OOV words

The uncorrected error rate was less than 0.5% for each method. The average error rates for MultiTap, Vision TiltText, T9, SHRIMP were 7.2%, 14.1%, 2.8% and 2.4% respectively. The overall error rate [23] of SHRIMP was significantly lower than that of MultiTap (p = 0.017). There was no significant difference in error rate between SHRIMP and DBD (p = 0.76). The error rate difference between SHRIMP and Vision TiltText was also significant (p < 0.001). The error rate difference between Vision TiltText and MultiTap was also significant (p = 0.04). This result agrees with previously reported error rates in similar tasks [20, 21].

Users in general have no trouble understanding the working mechanism of SHRIMP. Initially, many users tended to add more motion gestures than necessary, trying to enter a few characters with motion in each word. "I just want to verify whether the system works exactly as you described" one user explained. After they confirmed that adding motion gestures truly worked in SHRIMP, they started to trust the system. They typed familiar words without motion and used motion gestures when they were not sure whether a word, such as "`jedi`", was in the dictionary.

After finishing the study, the users generally felt positive about SHRIMP. They expressed their desire to switch from MultiTap to alternative input methods such as SHRIMP if they were available on their cell phone. Sample comments included -

"I found [SHRIMP based] predictive text input [effective] due to the reduced number of button presses".

"T9 + Tilt was pretty handy, if just b/c it's more interactive yet efficient with common words." "I found it convenient to have dictionary based input methods".

"SHRIMP is easy to learn, an improvement on T9, and resolve the problem of T9 when facing names or other special words which do not exist in the T9 lib".

"It's more efficient to remember the spatial location pattern in T9 and SHRIMP, 'the' is a big left arrow and 'you' is a small up arrow [on the keypad]".

Users also discovered some usability problems in the current SHRIMP prototype. One user complained that sometimes SHRIMP was not fast enough to follow her typing. She had to wait about half a second for the screen to refresh while she was typing a long word. Users also suggested that in addition to tactile feedback, some kind of visual feedback should be added when using the motion gesture.

FUTURE WORK

Our current study is only an initial step towards having a full understanding of SHRIMP. Enhancements such as adding new words to dictionary and richer visual/audio feedback can be made to the current prototype. A longitudinal study can be conducted to figure out the learning curve and expert performance of SHRIMP. A human performance model can be built to estimate its theoretical performance; of course, such a model will depend on a more accurate estimation of the distribution of words with encoding collision and OOV words in mobile environments.

CONCLUSIONS

Dictionary-based disambiguation (DBD) is a popular solution for text entry on mobile phone keypad, but it suffers from two problems: the resolution of collision (two or more words sharing the same key code) and entering out-of-vocabulary (OOV) words. Our analysis shows that both types of problems are quite frequently encountered. SHRIMP (Small handheld rapid input with motion and prediction) is a system and method that address these two problems by integrating DBD with camera based motion sensing. It enables the user to express preference through a tilt or move gesture. SHRIMP runs on camera phones equipped with a standard 12-key keypad. SHRIMP maintains the speed advantage of DBD driven predictive text input while overcoming the collision and OOV problems seamlessly without mode switching. By coupling a motion gesture with the action of typing the first character, SHRIMP can reduce encoding collision by more than 50%. By coupling two motion gestures, one with typing the first character and the other with typing the last character, SHRIMP can eliminate almost all encoding collisions.

An initial empirical user study showed that users can easily understand and learn SHRIMP with less than 10 minutes of practice. The text entry speed of SHRIMP (12.1 wpm) was significantly faster than that of MultiTap (7.64 wpm). The study also showed that SHRIMP can handle OOV words much faster than a traditional DBD method.

The SHRIMP concept is not limited to camera phone based motion sensing - this paper also contributes to the understanding of text entry based on ambiguous input in general. We unified the representation of action-based disambiguation and dictionary based disambiguation under the regular expression matching framework. The paradigm for SHRIMP can be applied to other multi-model input systems such as chording [19], accelerometer based tilting [20] and Nintendo Wiimote to achieve faster speed with a shorter learning curve.

SHRIMP has been implemented on unmodified, off the shelf Motorola V710 and Motorola RAZR V3 camera phones. SHRIMP is open source software released under BSD license. The current implementation can be downloaded from *http://bid.berkeley.edu/projects/shrimp*. We hope SHRIMP can inspire commercial implementations that change how people enter text on mobile phones in everyday life.

ACKNOWLEDGMENTS
The authors would like to thank David Nguyen, Ana Chang, Feng Tian, Lora Oehlberg, Andy Carle, Simon Tan and Antti Oulasvirta for their support in running the user study as well as authoring of the paper. We also thank the anonymous reviewers for their constructive feedback.

REFERENCES

1. Aho, A., Sethi, R., Ullman, J., Compilers: Principles, Techniques, and Tools, Addison Wesley Publishing, ISBN 0201100886.
2. Dunlop, M., and Crossan, A. Predictive Text Entry Methods for Mobile Phones. Personal Technologies, 2000.
3. Dunlop, M., Taylor, F., Tactile Feedback for Predictive Text Entry, In Proc. of CHI 2009.
4. Gong, J., Tarasewich, P., Alphabetically Constrained Keypad Designs for Text Entry on Mobile Devices, In Proc. of CHI 2005.
5. Grover, D.L., King, M.T., and Kushler, C. A. Patent No. US5818437, Reduced keyboard Disambiguating Computer. Tegic Communications, Inc., Seattle (1998).
6. How, J., Kan, M.Y., Optimizing Predictive text entry for short message service on mobile phones. In Proc. of HCII 2005.
7. Ingmarsson, M., Dinka, D., Zhai, S., TNT: a Numeric Keypad based Text Input Method. In Proc of CHI 2004.
8. International Telecommunication Union, Arrangement of Digits, Letters and Symbols on Telephones and Other Devices that can be used for gaining access to a Telephone Network, ITU Recommendation E.161. 1993.
9. International Telecommunication Union: Worldwide mobile cellular subscribers to reach 4 billion mark late 2008 http://www.itu.int/newsroom/press_releases/2008/29.html
10. James, C.L. and Reischel, K.M., Text input for mobile devices: Comparing model prediction to actual performance. In Proc. of CHI 2001.
11. Jansen, B., Zhang, M., el al., Twitter power: Tweets as electronic word of mouth, Journal of the American Society for Information Science and Technology, 2009.
12. Jelinek, F., Statistical methods for speech recognition, MIT Press, Cambridge, MA, 1998
13. Kucera, H. and Francis, W. N. Computational Analysis of Present-Day American English. Providence, Rhode Island: Brown University Press, 1967.
14. MacKenzie, I.S., Kober, H., et al., E. LetterWise: Prefix-based disambiguation for mobile text input. In Proc. of UIST 2001
15. Mayzner, M. S., and Tresselt, M. E. Table of single-letter and digram frequency counts for various word-length and letter-position combinations, Psychonomic Monograph Supplements 1 (1965), 13-32.
16. Rau, H., Skiena, S., Dialing for Documents: an Experiment in Information Theory, UIST 1994.
17. Silfverberg, M., MacKenzie, I.S., Korhonen, P. Predicting Text Entry Speed on Mobile Phones, In Proc. of CHI 2000.
18. Smith, S., Goodwin, N., Alphabetic Data Entry Via the Touch-Tone Pad: A Comment, HUMAN FACTORS, The Mitre Corporation, 1971, 13(2) P.p 189-190.
19. Wigdor, D., Balakrishnan, R., A Comparison of Consecutive and Concurrent Input Text Entry Techniques for Mobile Phones, in Proc. of CHI 2004.
20. Wigdor, D., Balakrishnan, R., TiltText: Using Tilt for Text Input to Mobile Phones. In Proc. of UIST 2003.
21. Wang, J, Zhai, S., Canny, J., Camera Phone Based Motion Sensing : Interaction Techniques, Applications and Performance Study. In Proc. of UIST 2006.
22. Why Book Means Cool http://www.languagehat.com/archives/002415.php
23. Wobbrock, J.O. and Myers, B.A. Analyzing the input stream for Character-level Errors in Unconstrained Text Entry Evaluations. In Proc of ACM ToCHI vol. 13 (4), 2006.
24. Wobbrock, J.O., Myers, B.A. and Chau, D.H. In-stroke Word Completion. In Proc. of UIST 2006.
25. Wobbrock, J.O., Chau, D.H. and Myers, B.A. An Alternative to Push, Press, and Tap-tap-tap: Gesturing on an Isometric Joystick for Mobile Phone Text Entry. In Proc. of CHI 2007.
26. Zhai, S., Hunter, M., Smith, B., The Metropolis Keyboard - an Exploration of Quantitative Techniques for Virtual Keyboard Design, In Proc. of *UIST 2001*.
27. Zhai, S. and Kristensson, P. Shorthand Writing on Stylus Keyboard. In Proc. of *CHI 2003*.