Fast text-entry on miniature mobile devices

F. E. Sandnes H. Thorkildssen A. Arvei J. O. Buverud

Faculty of Engineering
Oslo University College

Abstract

As mobile messaging devices are becoming smaller, for example wristwatches with Internet connectivity, there is less space for keys. Researchers have experimented with 3-key text input devices and come up with designs allowing text to be entered with as little as 4.23 keystrokes per character, but the strategies are hard to use. This paper demonstrates a dictionary-enhanced technique, which allows text to be entered using 2.16 keystroke per character. Four approaches are investigated — a ring-based method, a two-level tree-based method, a multi-tap method and a one-stroke method. Experiments are used to evaluate the usability of the approaches and the results confirm that there is a trade-off between usability and number of keystrokes per characters. The results show that the multi-ring approach requires less concentration from the user than previous strategies while text is entered with just 3.2 keystrokes per character.

1 Mobile text entry

Asynchronous messaging (SMS) using mobile devices has become an important means of written communication in Europe and is also popular in America and Asia.

Physical limitations and interaction controls

Common to all mobile devices is that they are small in physical dimensions and weight. The smaller and lighter a device is the more mobile it is and thus more desirable to the users. However, the small size comes at the expense of small displays for visual feedback and less room for interaction controls.

Controls vary from sophisticated touch sensitive displays accepting handwriting [3], voice activated input, jog-wheels (Sony), tilt-technology [11], wearable finger-rings [2] and traditional keys. Quite a few studies have studied virtual keyboards and keyboard layouts [7]. Virtual keyboards are usually implemented with touch sensitive displays, where the keyboard is drawn on the display. Virtual keyboards usually represent all the characters of the alphabet and the research has been focused towards layouts that results in faster text entry rates and models for evaluating keyboard layouts [7].

Simple physical keys have an extra appeal as they are cheap to manufacture and represents an interaction paradigm most users are familiar and comfortable

with. This paper addresses key controls, but the techniques described herein also extend to other interaction control technologies such as jog-wheels.

The limited surface area offered by the device only provides space for a few keys. There is also a trade-off between the number of keys and the size of the keys. Smaller keys are harder to use than larger keys and result in higher error rates as incorrect keys are more easily pressed accidentally. It is rarely room for full size keyboards, or even reduced-size keyboards. In general, mobile devices have fewer keys than there are characters in the alphabet. For example, mobile phones are usually equipped with standard phone dial keyboards comprising of 12 keys of which 8-keys are labelled for text entry in addition to a space key, and wristwatches usually have two or three keys or buttons dedicated to text entry purposes.

Text entry on telephone dial keyboards

Phone dial keyboards have a standard layout, where each key is labelled with three or four characters, e.g. key 2 is labelled with "a", "b" and "c", key 3 is labelled with "d", "e" and "f" and so forth. Mobile text-entry on these keyboards is classified into four categories. Multi-tap, dual-tap, chord and dictionary based. With the multi-tap technique the user hits a key repeatedly to cycle through characters labelled on the key to retrieve the desired character. With the two key technique [7] a character is selected by two key-strokes. The first character is used to select the character category, say "abc" and the second keystroke is used to select a character within this group. Chord techniques are based on pressing several keys simultaneously. One finger is used to press the key with the desired lettering and another finger is used to select the specific character within the group.

Dictionary based text-entry techniques [1, 5, 14] have won wide commercial success due to their ease of use and fast text entry speeds. The marked is dominated by the T9 system patented by Tegic, which is licensed by the major handset manufactures. Dictionary based text-entry allows text to be entered using close to one keystroke per character on phone keyboards. Approximately 94% of all words in the English dictionary are unambiguous [15] while the remaining 6% are ambiguous and the user must resolve the ambiguity manually by selecting the desired word from a list. Dictionary based systems are also employed in conjunction with touch sensitive displays (in particular the Octave text entry system by e-acute).

Three key text entry

Two and three key text entry systems first appeared on arcade game machines in the 70'ties, where users had to enter their names on high score lists using a joystick. The user would pull the joystick to the left or right to cycle to the alphabet (wheel-of-letters), or use rotator keys and press a select button to select the desired character — a method known as date stamp technique, after the method of setting the right date on an office stamp. Databank wristwatches followed in the 80'ies and numerous consumer electronics products come equipped with simple two or three key text-entry systems.

MacKenzie [10] has studied the date stamp approach at great detail and found that such text entry systems on average require 10.66 key-strokes per

character for normal English text. By making minor optimisations the number of keystrokes per character can be reduced to 6.45. Such text-entry systems suffice for occasional use but are slow and annoying to use on a regular basis. The date-stamp technique also requires high concentration from the user who must track the position of the wheel of letters. Attempts at improving this have included using a dynamic wheel-of-letters, where letter di-grams are used to arrange the letters in a distance to the cursor with respect to their probability of occurring after the previously entered character. Such techniques have resulted in systems requiring only 4.23 keystrokes per character. However, experiments have shown that these systems are hard to use because they require high concentration from the user navigating a constantly changing wheel-of-letter.

Another three key text entry technique was introduced by Raghunath and Narayanaswami [12], which we have chosen to term the dual-ring approach. Their strategy, implemented on a wristwatch, consisted of splitting the alphabet into two and presenting the alphabet as two rings. The outer ring contained the characters from "a" to "p" and the inner ring the characters from "q" to "z". The user could toggle between the two rings and then rotate the rings to obtain the desired key. I.e. one key is used to toggle between the two rings, one key is used to cycle forward in the rings and the final key is used to select a letter.

Three-key text entry devices are attractive for a number of reasons. First, miniature devices with a very small surface area will in some instances not have physical space for more than three keys. Second, three key devices are cheaper to manufacture than devices with more keys. Third, fingers need not be moved between keys as three fingers can be fixed onto the three keys [6]. Time is thus saved avoiding unnecessary physical movement and visual inspection of finger positions relative to key positions. Fourth, three key devices are non-intimidating and do not come across as technically complex to users with moderate computer literacy.

Dictionary based text entry

A text entry device can be defined as $T(H, A)$, where H is a set of keys $h_1, h_2, .., h_k$, k is the number of available keys, A is the alphabet of characters $a_1, a_2, .., a_L$ and L is the number of characters in the alphabet.

We define a word w as a sequence of characters $a_x, a_{x+1}, .., a_Z, \in A$, where Z is the length of the word. Further, we define a dictionary $D = \{w_1, w_2, .., w_P\}$ as the set of all valid words, i.e. valid character sequences that occur in the source language, where P is the number of dictionary entries. A key-map is defined as $M = \{m_1, m_2, .., m_L\}$ where $m_i \in \{1..k\}$ and m_i indicates which key the character a_i is associated with. Next, a word mapping function can be defined as $w' = Q(A, M, w)$, where $w \in A$ is the original word in its original alphabet, and $w' \in M$ the word represented in terms of its key-map. Given a keystroke sequence $S \in \{k_1, k_2, ...\}$ we can define the inverse mapping function $U = Q^{-1}(A, M, S)$, which given a keystroke sequence S returns the set of matching words U. If $|U| = 1$ then there are no ambiguities, if $|U| = 0$ then there are no words matching the keystroke sequence (probably due to a typing error) and if $|U| > 1$ then U contains a list of ambiguous words which must be resolved. We define all the ambiguous words in the dictionary D as V.

Resolving ambiguities

Mobile text entry techniques can be evaluated and compared in terms of key strokes per character (KSPC) and one can measure this in terms of the minimum number of keystrokes K_{min}, the maximum number of characters K_{max} and the mean number of characters $\overline{K}$. The total number of keystrokes K can be decomposed into the following for dictionary based text entry techniques:

$$K = K^a + K^b \tag{1}$$

where K^a is the number of key strokes needed to compose the word[1], and K^b is the number of keystrokes needed to select the desired word when there are several ambiguous entries. Clearly, K^b is 0 for all words where there are no ambiguities, and $K^b > 0$ for all ambiguous words. For T9 systems there are rarely more than 8 ambiguous words, and K^b is usually less than 3 (assuming the user is using the interface correctly). Note that this paper distinguishes between K^a and K^b. For a general discussion on KSPC see [13, 9, 4] and [8].

Alphabet partitioning

For text entry techniques where the alphabet size is larger than the number of keys ($k < L$) and each key is assigned a group of letters, the way the partitions are devised strongly affects the rate of ambiguities. Generally, an alphabet can be grouped into the partitions c_1, c_2, .., c_N with the sizes C_1, C_1, .., C_N correspondingly, where N is the number of partitions. There are many ways to arrange the partitions. However, the number of configurations can be reduced by only considering alphabetically ordered character sequences. Users are familiar with alphabetically ordered lists of letters, and this letter-arrangement makes the systems easy and quick to learn. The optimal key mapping M can be defined as the partitioning configuration that leads to the smallest numbers of ambiguities.

2 Dictionary based three-key text entry

This paper addresses dictionary-enhanced mobile text entry with three keys. Five novel text entry techniques are presented. We term these methods multi-ring, tree-based, multi-tap, one-stroke and dictionary-lookup.

Method #1 : Multi ring text entry

In the multi ring approach the alphabet is split into the N partitions. In this discussion the character partitions found on standard mobile handsets are used, i.e. “abc def ghi jkl mno pqrs tuv wxyz”. This configuration would allow the standard T9 text-entry system to be used as a sub-system.

To enter a character the user cycles through the partitions using the left and the right keys. When the partition containing the desired letter is highlighted the partition is selected using the select key. The cursor is moved back to the origin and the process is repeated for subsequent characters.

The following example illustrates how a user types the word “code”

[1]Note that this is a label and not the power function.

```
 1(2) [abc]
 2(3)*[abc]
 3(3)*[def]
 4(3)*[ghi]
 5(3)*[jkl]
 6(2)*[mno]
 7(3)**[abc]
 8(2)**[def]
 7(3)***[abc]
 8(2)***[def]
 9(1)****[abc]
10(2)****[_]
11( )code [abc]
```

To enter the character "c" the user simply presses the middle key to select the default group "abc" since it contains the "c". Next, to enter the "o" the user must press the right key four times to reach the group "mno" containing the "o" followed by the middle key to select it. Then, to enter the d and the "e", which are both in the second group, the user must press the left key to reach the "def" group followed by the middle key to select it — twice. Finally to indicate the end-of-word, the user presses the left key to reach the space category followed by the middle key to select it. This keystroke sequence is unambiguous and the word "code" + space is displayed. A total of 10 keystrokes are needed to enter this expression.

Since the T9 dictionary system operates with a large number of partitions (eight) there are only a small fraction of ambiguous keystroke sequences. The ambiguous sequences have a limited number of alternatives and a simple strategy for resolving the ambiguities is sufficient. One such strategy is to scroll to through the list of words using the left and the right key and then select the desired word using the middle key. For example, say the user wants to enter the word "car", which is ambiguous.

```
 1(2) [abc]
 2(2) *[abc]
 3(1) **[_]
 4(1) **[wxyz]
 6(1) **[tuv]
 7(2) **[pqrs]
 8(1) ***[abc]
 9(2) ***[_]
10(3) [<- bar (1/3) ->]
11(3) [<- cap (2/3) ->]
12(2) [<- car (3/3) ->]
13( ) [abc]
```

Steps 1 to 9 are used to enter the keys assigned the characters "car" + space in the same manner as in the previous example. Then, at step 10 the text entry interface goes into an ambiguity-resolving mode. In this example the user scrolls through all the words matching the keystroke sequence using the right key ("bar", "cap" and "car"). Once the desired word "car" is displayed it is selected by pressing the middle key. In this example 12 keystrokes were needed to enter the 4 letter expression "car" and space. However, this could be achieved with just 11 keystrokes if the user used the left key instead of the right key at step 10.

The minimum number of keystrokes needed to retrieve a character is $K^a_{min} = 1$ if the character is either "a", "b" or "c". The maximum number of keystrokes K^a_{max} required to retrieve a character without errors is 5. Generally, given N partitions then

$$K^a_{max} = \left\lfloor \frac{N}{2} \right\rfloor + 1 \tag{2}$$

Further, the average number of keystrokes $\overline{K^a}$ is

$$\overline{K^a} = \sum_{i=1}^{N} (\delta(i, N) + 1) f(c_i) \tag{3}$$

where $\delta(i, N)$ is a distance function indicating the distance to category i with N categories defined as

$$\delta(i, L) = \begin{cases} i, & i < L/2 \\ L - i, & i \geq L/2 \end{cases} \tag{4}$$

and $f(c)$ is the combined frequency for the characters in category c for the given language. For the partition described above $\overline{K^a}$ is 3.04 keystrokes per character.

Note that in addition to the eight partitions a ninth partition is added consisting of only the space character, which is usually used as a word delimiter in normal texts. This is placed in the last category so that it can be reached with just one keystroke (left). In our implementation periods are retrieved by entering two space characters consecutively.

The average number of keystrokes per character associated with resolving ambiguous words is given by:

$$\overline{K^b} = \sum_{w \in V} \frac{R(w) f(w)}{|w|} \tag{5}$$

where $R(w)$ is the number of keystrokes needed to select the desired word from the list, $f(w)$ the frequency of occurrence for the word w in the language. Further,

$$K^b_{max} = \max_{w \in D} \frac{R(w)}{|w|} \tag{6}$$

For this strategy $\overline{K^b}$ is approximately 0.16, K_{max} is 0.5 and K^b_{min} is 0. These numbers where computed using Kilgariff's reference word frequency list and Equations (5) and (6).

Method #2 : Tree-based text entry

The tree-based text entry approach consists of entering a letter by walking a three-way tree in two steps. In the first step the users choose between three major partitions using one of the three available keys and in the second step one of the three sub-partitions is chosen using one of the three keys. The two keystrokes provides sufficient information to determine the category of the desired character. The partitioning scheme used for the multi ring approach is also used for the tree-based approach. The eight categories and space can be organised into a perfectly balanced three-way tree with two levels.

For example, the user enters the word "code" and space as follows using the tree-based approach:

```
 1(1) [abc#def#ghi jkl#mno#pqrs tuv#wxyz#_]
 2(1) [abc def ghi ----------------------]
 3(2) *[abc#def#ghi jkl#mno#pqrs tuv#wxyz#_]
 4(2) *[----------- jkl mno pqrs -----------]
 5(1) **[abc#def#ghi jkl#mno#pqrs tuv#wxyz#_]
 6(2) **[abc def ghi ----------------------]
 7(1) ***[abc#def#ghi jkl#mno#pqrs tuv#wxyz#_]
 8(2) ***[abc def ghi ----------------------]
 9(3) ****[abc#def#ghi jkl#mno#pqrs tuv#wxyz#_]
10(3) ****[abc def ghi ----------------------]
11( ) code [abc#def#ghi jkl#mno#pqrs tuv#wxyz#_]
```

The first character "c" is located in the first part of the first group, and the left key is therefore pressed twice. The first keystroke selects the group "abc#def#ghi", and the second keystroke selects the "abc" partition. Next, the "o" is located in the middle part of the middle group and thus the middle key is pressed twice. Then the "d" and the "e" are located in the second part of the first group which are selected by pressing the left key followed by the middle key twice. Finally, end-of-word is indicated by selecting the space character. Space is located in the right part of the rightmost group, which is accessed by pressing the right key twice. This keystroke sequence in unambiguous and the word "code" and space are displayed. Thus, the four letter word "code" + space is entered using 10 keystrokes.

Ambiguities are handled the same way they are handled in the MultiRing approach. The values for K^b are therefore the same.

The advantage of the tree-based approach is that only two keystrokes are needed to retrieve letters, i.e. $K^a_{min} = K^a_{max} = \overline{K^a} = 2$.

Method #3: Multi-tap

In this strategy the characters are arranged into eight groups in a similar manner to what can be found on most telephone dials. Quartets of categories are assigned to the keys and the user accesses a group by tapping the assigned key repeatedly until the desired group is highlighted. I.e. key-1 is assigned "abc", "def", "ghi" and "jkl", key-2 is assigned "mno", "pqrs", "tuv" and "wxyz" and key-3 is assigned the space character and functions as a break key. For example, the word "do" is entered as follows:

```
 1(1) []
 2(1) [<ABC> def ghi jkl]
 2(3) [ abc <DEF>ghi jkl]d
 3(4) [<MNO>pqrs tuv wxyz]d
 3(4) [break, space]dm
 3( ) []do_
```

The user first hits key-1 twice to select the "def" group, then key-2 to select the "mno" group and finally key-3 twice to select the space.

Clearly, $K^a_{min} = 1$, $K^a_{max} = 4$ and $\overline{K^a} = 2.17$. Note that these estimates do not include the overhead of the break key. Ambiguities are resolved in a similar manner to the previous approaches, namely $\overline{K^b} = 0.16$ and $K^b_{max} = 0.5$.

Method #4: One-stroke text entry

In the one-stroke approach the user is presented with three partitions - two comprising of alphabetic letters and the third of the space character. The optimal partitioning for English is "abcdefghijklm" and "nopqrstuvwxyz" (determined

by brute-force search). The user enters a word by selecting one of the tree partitions using the left and the middle key and presses the right key when reaching the end of the word to indicate space. For example, the 13 letter word "approximately" and space are entered as follows using only 14 keystrokes.

```
 1(1) [abcdefghijklm nopqurstuvwxyz _]
 2(2) *[abcdefghijklm nopqurstuvwxyz _]

 ... then keys 2, 2, 2, 2, 1, 1, 1, 2, 1, 1, 2 ...

14(3) ************[abcdefghijklm nopqurstuvwxyz _]
15( ) approximately [abcdefghijklm nopqurstuvwxyz _]
```

First, the left key is pressed since it is assigned the character "a". Next, the middle key is pressed five times as it is assigned the letters "pprox". Subsequently, the keys, left, left, left, middle, left, left middle are pressed for the characters "imately" and finally the right key is pressed to retrieve the space character. Only 14 keystrokes are needed to enter the 14-character word including space.

However, with only two character partitions only 7.5 % of the words are unambiguous, while the remaining words are ambiguous. Further, the number of ambiguous words in each ambiguous instance is much larger than for T9. The worst case is 202 alternatives[2]. Clearly, choosing a word from such a long list in a linear fashion is extremely time-consuming and tedious. Therefore, a quicker and intuitive strategy was adopted which is not concentration intensive. In this strategy, the ambiguous words are resolved by fixing the letters one at a time until the desired word is identified. First, the user selects the first letter in the word. Then the user selects the second letter and so forth. Usually, only just a few selections are needed to correctly identify a word from a large list. The user employs the left and the right keys to choose the desired character and then the middle button to select the chosen character. For example, to enter the word home

```
 1(1) [abcdefghijklm nopqurstuvwxyz _]
 2(2) *[abcdefghijklm nopqurstuvwxyz _]
 3(1) **[abcdefghijklm nopqurstuvwxyz _]
 4(1) ***[abcdefghijklm nopqurstuvwxyz _]
 5(3) ****[abcdefghijklm nopqurstuvwxyz _]
 6(1) [<- (a)nal ->]
 7(1) [<- (m)ock ->]
 8(1) [<- (l)oad ->]
 9(1) [<- (j)oel ->]
10(1) [<- (i)nca ->]
11(2) [<- (h)off ->]
12(2) [<- h(o)ff ->]
13(1) [<- ho(f)f ->]
14(2) [<- ho(m)e ->]
15( ) home [abcdefghijklm nopqurstuvwxyz _]
```

First the keys left, middle, left and left are pressed to enter the letter sequence "home", followed by the right button to mark an end-of-word (space). The keystroke combination is ambiguous and these ambiguities must be resolved manually by the user (see step 6 and onwards). The user first locates the first character "h" by pressing the left key five times and then the middle key to select it. The interface then moves to the second character "o". The user presses the

[2]This number is dictionary dependent.

middle button to accept the "o". In the last step the user must determine the third character and presses the left button to locate the "m" and then the middle button to select it. The ambiguity is then completely resolved and the word "home" followed by a space is displayed. A total of 14 keystrokes are needed to enter this expression.

Clearly, with this strategy $K^a_{min} = K^a_{max} = \overline{K^a} = 1$ which is the same as a full size keyboard, but without the need to move fingers across a keyboard.

Using Kilgariff's word frequency list and equations (5) and (6) we find that $\overline{K^b}$ is 1.97, K^b_{max} is 4 and K^b_{min} is 0.

3 Method

Subjects

Twelve volunteer subjects (nine male and three female) participated in the experiment, and were mostly students at Oslo University College. Nine subjects were in their early twenties and three were around 50 years old. The subjects reported spending on average 24 hours a week in front of a computer. All subjects except one reported mastering touch-typing on conventional full-size QWERTY keyboards, and all apart from two subjects reported being fast typists on mobile phones. All subjects reporting sending one or more SMS message a week. One subject reported being visually impaired.

Apparatus

The experiment was carried out using several Internet enabled workstations. The workstations consist of standard monitor displays, full-size QWERTY keyboards and mouse. The different text entry techniques were implemented as part of a modular custom applet framework, running in a web-browser. All the keystrokes entered by the users are recorded and continuously transmitted from the text-entry applet to a server application, which logs the keyboard events in a database. These entries were used in the subsequent analysis.

The applets where configured to accept key events from the left, down and right arrow keys representing the keys 1, 2 and 3 respectively.

The dictionary used in the experiments is adapted from the wordlist that is found on most UNIX systems.

The source text was displayed on the same web page as the applet, but in a separate frame. The user could conveniently browse and navigate the text. The text comprised of 40 quotations presented in groups of four.

Procedure

The trials were performed individually and in isolation. The subjects were first informed of the proceedings and purpose of the overall experiment. Then they were instructed in the different text entry techniques and five minutes were set aside for practice, using the text-entry software in training mode. To control the size of the experiment only the multi-ring (method #1), tree-based (method #2) and one-stroke (method #4) methods were included. The subjects where allowed 15 minutes for each of the text entry strategies. The order of the trials was varied across the subjects to reduce bias.

Table 1: The mean time (in seconds) between each consecutive keystroke.

Subject	multi-ring	tree	one-stroke
subject 1	0.9	1.6	1.0
subject 2	1.2	2.6	1.2
subject 3	1.8	2.7	1.8
subject 4	1.1	2.6	1.3
mean	1.25	2.375	1.325

Table 2: The subjects' impressionistic evaluation of the text-entry strategies.

Method	tree	one-stroke	isolation
multi-ring	0.27	0.42	0.69
tree		0.67	0.5
one-stroke			0.77

At the end of the experiments the subjects were asked to complete a questionnaire asking for demographic information and subjective evaluations of the text entry techniques.

4 Results and discussion

Table 1 shows the result of the measurements acquired during the typing experiments. The table lists the mean number of seconds between consecutive keystrokes with the three techniques for the four subjects[3]. Outliers in the measurements are removed (i.e. delays with a duration of more than 10 seconds). The last row shows the overall mean for the tree techniques.

According to these results the methods can be ranked as follows: multi-ring, one-stroke and tree-based, since the average delay between consecutive keystrokes represents the time needed by the user to make a decision about which button to press. Clearly, the multi-ring is associated with the shortest delay of 1.25 seconds on average, closely followed by 1.325 seconds for the one-stroke technique and finally 2.375 seconds for the tree-based strategy — although the delays are nearly identical for the two methods for subject 2 and 3. Thus, it takes the user a shorter time to select which button to press when using the multi-ring approach, and the longest time for the tree-based approach.

Clearly, it is easier, in terms of concentration, to select a neighbour from a list by cycling through its items rather than making selections directly from a list — especially when these selections are made in a hierarchal manner across several steps.

Table 2 shows the subjects' impressionistic evaluation of the text-entry strategies based on means computed from the questionnaire. Columns two and three show how the subjects rate the three methods comparatively. A value closer to 1 indicates that the subjects prefer the method specified by the column and a value closer to 0 indicates that the subject prefers the method specified by the

[3]Only four subjects were recorded due to technical difficulties with the keystroke recording module.

Table 3: Estimated text entry rates for untrained users.

Method	delay	$\overline{K^a}$	$\overline{K^b}$	$\overline{K}$	time /char	word /min
Dat-stp	≈1.1	7.77	0.00	7.77	8.54	1.4
Rag.Nar.	≈1.1	7.19	0.00	7.19	7.91	1.5
McKenz.	≈2.6	4.23	0.00	4.23	10.99	1.1
M-ring.	1.1	3.04	0.16	3.20	3.52	3.4
Tree	2.6	2.00	0.16	2.16	5.62	2.1
M-tap	≈1.1	2.17	0.16	2.33	2.56	4.7
1-stroke	1.3	1.00	1.97	2.97	3.86	3.1
Qwerty	≈1.0	1.00	0.00	0.00	0.20	12.0

row. Column four indicates how the subjects rate the method in isolation, where a value closer to 1 indicates that the subjects like the method and a value closer to zero indicates that the subjects dislike the method.

According to these results the least favourable strategy is the tree-based method as it is ranked the worst in isolation (0.5) and also ranked worse than both the multi-ring and the one-stroke approaches as multi-ring is preferred with a weight of 0.73 and the one-stroke method is preferred with a weight of 0.67. When comparing multi-ring and one-stroke then one-stroke is preferred when looking at the isolated evaluation (0.77), but when looking at the comparison of the two then the multi-ring techniques comes out the best with a weight of 0.58. Thus, impressionistically the methods are ranked as follows: multi-ring, one-stroke and tree-based. This also corresponds to the measurement results presented in Table 1.

Table 3 uses the findings in this paper to estimate the number of words per minute feasible with the different approaches. The table includes the strategies presented within this paper as well as results for the date-stamp approach, Raghunath and Narayanaswami's approach, MacKenzie's approach and full-size QWERTY for comparison. For the date-stamp approach the delay measurement for the multi-ring is used (easy task), for MacKenzie's method the tree-based delay is used (hard task) and for QWERTY it is assumed the typist is untrained with an inter-keystroke delay of 1 second. The first column lists the methods, the second gives the most appropriate inter-keystroke delay for the methods, column three and four lists $\overline{K^a}$ and $\overline{K^b}$ for the methods and the fifth column lists the sum of these, namely K. The sixth column lists the number of seconds per character computed by multiplying the inter-keystroke delay with the average number of keystrokes per character. The last column lists the estimated words per second, where it is assumed that the mean word length is 5 characters. When the measured typing delay is combined with the theoretical keystrokes per character the ranking of the methods remains unchanged. Clearly, with the multi-ring approach one should be able to type 3.4 words per minutes, followed by 3.1 words per minute with the one-stroke approach and only 2.1 words per minute with the tree-based method.

5 Summary

Techniques for dictionary enhanced text input on low-cost miniature devices with three keys are proposed. The strategies proposed result in faster text-entry rates than what has previously been reported in the literature for three-key devices. In particular, the multi-ring approach allows text to be typed with 3.2 keystrokes per character while being very pleasant and easy to use and the three-based strategy allows text to be entered with just 2.16 keystrokes per character. The techniques are simple and cheap to implement, and quick and easy to learn and use. Dictionary enhanced text entry strategies are feasible especially as miniature devices are capable of storing large dictionaries at low cost.

References

[1] Arnott. Probabilistic character disambiguation for reduced keyboards using small text samples. *AAC augmentative and alternative communication*, 8, 1992.

[2] M. Fukumoto and Y. Tonomura. Body coupled fingering: wireless wearable keyboard. In *ACM Proceedings of CHI97, pp147-154*, 1997.

[3] D. Goldberg and C. Richardson. Touch typing with a stylus. In *Proceedings of ACM InterchiŠ93*, pages 80–87, 1993.

[4] C. L. James and K. M. Reischel. Text input for mobile devices: Comparing model prediction to actual performance. In *ACM proceedings of CHI 2001*, pages 365–371, 2001.

[5] M. T. King. Justtype. tm. - efficient communication with eight keys. In *Proceedings of the RESNA 95ÍAnnual conference, Vancouver, BC, Canada*, 1995.

[6] J. Lehikoinen and M. Roykee. N-fingers: A finger-based interaction technique for wearable computers. *Interacting with Computers*, 13:601–625, 2001.

[7] I. MacKenzie and R. W. Soukoreff. Text entry for mobile computing: models and methods, theory and practice. *Human Computer Interaction*, 17(2):147–198, 2002.

[8] I. S. MacKenzie. Kspc (keystrokes per character) as a characteristic of text entry techniques. In *Proceedings of HCI Mobile 2002*, 2002.

[9] I. S. MacKenzie and S. R. W. A character-level error analysis technique for evaluating text entry methods. In *ACM Proceedings of NordCHIŠ01*, pages 243–246, 2001.

[10] S. MacKenzie. Mobile text entry using three keys. In *ACM proceedings of NordCHIŠ02*, pages pp27–34, 2002.

[11] K. Partridge, S. Chatterjee, and R. Want. Tilttype: Accelerometer-supported text entry for very small devices. In *Proceedings of AMC CHIŠ01*, volume 4, pages pp201–204, 2001.

[12] M. T. Raghunath and C. Narayanaswami. User interfaces for applications on a wrist watch. *Personal and Ubiquitous Computing*, 6:17–30, 2002.

[13] M. Silfverberg, I. S. MacKenzie, and P. Korhonen. Predicting text entry speed on mobile phones. In *ACM proceedings CHIŠ2000*, volume 1, pages 9–16, 2000.

[14] S. L. Smith. Alphabetic data entry via the touch tone pad: a comment. *Human factors*, 13(2):189–190, 1971.

[15] I. H. Witten. *Principles of computer speech*. Academic Press, 1982.