Modelling Text Input Strategies for Miniature Portable Devices with Finite State Automata

Frode Eika Sandnes

Faculty of Engineering
Oslo University College

Abstract

In this paper a methodology for representing text-input strategies for miniature and mobile devices is proposed. The methodology is based on representing text-input strategies as graphs. Graph representations allow different static mobile text-input strategies to be represented in a uniform manner. Further, different strategies are easily compared as the graph representation allows various characteristics to be extracted. The methodology incorporates KSPC (KeyStrokes Per Character), checking for error recoverability and correctness. The methodology can be expanded to include additional evaluation measures and it is feasible to implement design-tool support.

1 Introduction

Controls on mobile devices vary from touch sensitive displays accepting handwriting [11], voice activated input, jog-wheels (Sony), tilt-technology [29], wearable finger-rings [10] and keys. One avenue of research addresses virtual keyboards and keyboard layout [26] implemented using touch sensitive displays, where the keyboard is drawn on the display. This paper addresses key controls, but the techniques described herein also extend to other interaction controls such as wheels or joysticks.

The limited surface area offered by a device may only provide 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 leads to higher error rates as incorrect keys are more easily pressed accidentally (Fitts' Law [9]). In general, mobile devices have fewer keys than there are characters in the alphabet.

The keys on numeric keypads are usually 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. A character is thus retrieved with anything from 1 to 5 keystrokes. With the two-key technique [26], a character is selected by two keystrokes. The first keystroke is used to select the character category, say "abc" and the second keystroke is used to select a character within this group. Two-keystrokes are required per character. 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 [15, 17, 20] involves speeding up the text-entry by the means of dictionaries. Tegic's T9 is one such widely used system.

Two and three key text entry systems first appeared on arcade game machines in the 70'ties, where users entered 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, or use rotator keys and press a select button to select the desired character — a method known as the date-stamp technique, after the method of setting the right date on an office stamp by cycling through the digits. Databank wristwatches followed in the 80'ies and numerous consumer electronics products have simple two or three key text-entry systems. MacKenzie [25] describes the date-stamp approach at great detail and proposes several optimisations.

Another three key text entry technique was introduced by Raghunath and Narayanaswami [30]. Their strategy, implemented on a wristwatch, consisted of splitting the alphabet into two and presenting the alphabet as two rings. 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 character.

Sandnes et al. [35] investigated three other text entry strategies for devices with three keys, namely multiring, tree-based and binary as well as dictionary accelerated versions of these techniques [34]. In the multi-ring approach characters are organised into groups, for example "abc", "def", "ghi" and so forth. First, the user cycles through the list of groups using a "left" and a "right" key and then selects the group containing the desired character. Then, the user cycles through the characters within the group. The tree-based strategy comprise of extracting the desired character from the 27-letter alphabet including space by using one of the tree keys to iteratively select one third of the characters. After three selections one character remains. In the binary approach the user retrieves the desired character by binary search.

Evreinova et al [8] has proposed a static text entry strategy using four keys intended for disabled user and Tamaka-Ishii et al [39] studied a four-key text dictionary enhanced entry strategy which employs a combination of disambiguation and prediction.

Five key text entry is has been around for over 100 years. The first five key text entry techniques were chord based keyboards originally used for mail-sorting applications (see the excellent survey of chord based text entry by Moyes [28]). Several studies addressing various aspects of five-key chord keyboards exist [12, 16, 28, 31, 37] and commercial chord keyboards include the Microwriter. Five-keys have also been proposed for use in stenograph typewriters [3].

The five keys allow 31 unique chords to be entered and thus allow the English alphabet to be addressed. Although allowing for fast text entry rates, such systems have a high learning threshold, although Gopher and Raij [12] claim that chording in principle is easier to learn than QWERTY touch typing. When Douglas Carl Engelbart invented the mouse in the 60'ies he originally visualized the mouse with five-keys allowing the users to type text using chords [1]. Some chord keyboards have more than five keys such as the Twiddler keyboard [22, 23]. Expert typists manage to enter text using one hand on the Twiddler keyboard at rates of up to 67 words per minute. An interesting hybrid of a three-key chord keyboard and numeric keypad allows one finger to select the desired letter group on the keypad and the other hand to disambiguate the letter using a chord concurrently and hence reduce the input delay.

Several strategies has been proposed for one-handed text-entry such as the ingenious half QWERTY keyboard [27]. The principle of the half QWERTY keyboard is to transfer

two-handed QWERTY typing skills using one hand on a special half keyboard. A totally different approach taken by Isokoski and Raisamo [13] is to enter text in graphical patterns controlled by device independent north, south, east and west movements to mimic the graphical shape one would handwrite a character. Another class of simple five-key text-entry systems can be found on computer game consoles, for example the Nintendo Advance. Characters are organised into a two-dimensional wraparound mesh. This mesh is navigated by using four navigation keys and select [4]. Three other five-key text entry techniques are investigated in [32].

Benefits of a graph based methodology

There are several benefits to introducing a graph-based methodology for modelling text entry strategies. First, the text entry strategies must be static in nature since graphs are static structures. Hence, no dynamic aspects can be modelled. There are several major challenges associated with dynamic and adaptive user interfaces from a usability point of view. Several studies on mobile text entry conclude that adaptive and dynamic text entry strategies are hard to learn and thus slow to use in practice although they are theoretically fast to use (See [4, 25]). Further, adaptive and dynamic strategies require feedback and cannot be used eyes-free.

Second, graph theory is well understood and widely used in computer science including human computer interaction [5], and a wide range of graph theoretic algorithms and graph analysis metrics exists, which can be applied to the evaluation and analysis of text- entry systems. Especially, graph algorithms used in networking and parallel and distributed processing are of interest (See fo r example [18]. Some of these possibilities are explored in subsequent sections.

Third, the graph notation allows automated verification and analysis of text-entry designs to be carried out by automatic tools. We have implemented a prototype text-entry engineering tool that allows the users to graphically design text entry strategies, easily compute statistics and characteristics of the design, evaluate correctness and quality of the designs and automatically generate a text entry java midlet capable of running on a mobile handset [36].

2 A graph based methodology

The purpose of this methodology is to identify weaknesses in text entry designs before one is committing to user tests, as testing on real people is an expensive and time-consuming activity. Note however that the methodology is not a substitute for user testing, and that testing is necessary in order to identify cognitive bottlenecks. It is intended to compliment other evaluation methods and the evaluation of other aspects of interaction such as analysis of cognitive difficulty [33], studies of interference [21], bimanual input [6] just to mention a few.

Scope

The methodology in this paper has two limitations, it is intended for static text entry strategies and it is intended for situations where there is a limited set of control signals or device stimuli. A static text entry strategy can be defined as one that does not change throughout its lifetime and that its response is predictable. For example, text entry prediction and disambiguation are two separate and important aspects of mobile text entry that are dynamic in nature.

A limited set of control signals means that the physical characteristics of a device has only a small number of ways in which users can interact with the device. For example, a desktop has a full size QWERTY keyboard with more than 100 keys, while a wristwatch may only have three keys for interaction. Due to the limited number of control signals, several signals must therefore be combined into a sequence in order for symbols to be produced or operations to be executed. Interaction strategies that apply a sufficient number of interaction signals has no need for sequential input and are therefore not relevant for this strategy.

Devices

In this study it is assumed that the device is of limited physical size and thus has a limited surface area. Further, it is assumed that the device has a form of visual, aural or tactile feedback and some interaction controls, such as keys. The interaction controls can be used to send interaction signals defined by the set $S = s_1, s_2, .., s_N$ where N is the total number of control signals. These controls are typically keys, and the devices would typically be one-hand devices, i.e. they are held in one hand and operated by the other hand. Note that the model is capable of representing two-handed operation as well. In this study devices with three keys are used for illustrative purposes.

Signals

Given a device with K keys, in this instance three keys, several categories of signals can be sent. First, single keystrokes can be used to send K different signals. Second, chording can be used to increase the number of possible signals. With K keys $2^K - 1$ signals can be sent and with three keys 7 different signals can be chorded. Third, keystroke duration can be used to express different signals — for example short taps and long or hold strokes. With K keys $2K$ signals can be sent when allowing for short and long taps. If chording and keystoke duration are combined a total of $(K^2 - 1)$ signals can be generated. For 3 key devices this would account to 14 different signals. Note however that a combination of chording and keystroke duration is difficult and would require practice.

In the proposed methodology single keystrokes are denoted by e_i where e_i indicates that key e_i is pressed, $e_i + e_j$ indicate a chord signal comprising of the keys e_i and e_j, and a short tap is denoted by $\dot{e_i}$ and a long tap is denoted by $\overline{e_i}$.

The alphabet

The alphabet is a set of symbols that the user needs in order to compose the desired texts. This study is restricted to languages using phonetic scripts represented by a limited set of symbols, for example most European languages such as English. Languages such as Chinese or Korean may need a totally different approach.

Editing

Advanced editing is included into the model by treating the editing commands as part of the character stream. In order for advanced editing to be incorporated into the model text entry strategy must therefore be equipped with editing symbols, for example BACKSPACE, FORWARD, BACKWARD, UP and DOWN, to mention a few. Further, case is included by providing TO-UPPERCASE and TO-LOWERCASE symbols or simply a CAPS-LOCK symbol. The particular editor is therefore responsible for interpreting these editing symbols that arrive in the symbol stream.

Modelling text entry strategies

A static text entry strategy can be modelled using a finite state machine modelled as a directed graph $G(V, E)$, where the vertices V represent text-entry states and the edges E represent transitions between these states. The states typically represent time- intervals when the device is waiting for input from a user while the users are determining their next move. A transition is triggered by a user signal, i.e. when a user presses a key, and the finite state machine moves into a new state. Transitional edges are therefore labeled with a signal label s_i indicating the user signal (or keystroke) that triggers the transition. A transition may also trigger an output signal. This is denoted using the notation $s_i : a_j$, where the signal s_i triggers the symbol a_j. For example "2:c" denotes that signal "2" produces the output "c", "3:" denotes that signal "3" leads to a state transition but no symbol is output, ":d" defines a default state transition that occur without a signal but produces the symbol "3" and finally ":" indicates a default state transition without an output symbol.

To simplify the diagram and increase readability end states are denoted by edges that do not point at other states. End states are therefore also easy to identify in a diagram. The graph should always comprise a start state or start vertex. In this paper start states are represented using a gray shaded vertex. A graph can only have one starting-state but may possess multiple end-states.

Given a test entry strategy defined using graph G, then the shortest path between the two states C_a and C_b is represented using $Path(G, C_a, C_b)$, and the length of this path is $]Path(G, C_a, C_b)]$. We also define an output state $C_{out}(a_i)$ as a state that comprise a outbound edge representing a transition that produces the output symbol a_i. Finally, the set of exit states C_{exit} are states without outbound edges.

3 Evaluating text-entry strategies

In this section evaluation criteria for mobile text-entry are explored, namely correctness, error recoverability and KSPC.

Correctness

We define the correctness of a text entry strategy to mean that it is both feasible and it has full coverage. A feasible strategy can be implemented within the constraints of the target device, and a strategy has full coverage if all the symbols of the alphabet can be output.

For a device to be feasible the following two criteria must be satisfied. First, the number of outbound edges or leaving transitions from a given state must not exceed the number of signals $]S]$ supported by the device. A simple graph traversal can be used to ensure that this constraint is satisfied. Second, each trigger of a state must be unique. If two triggers or more are identical then the design is ambiguous. This constraint can be verified by a simple graph traversal.

For a text entry strategy to cover the entire alphabet there must be at least one unique transition for each symbol of the alphabet. I.e. for each symbol in the alphabet there must be an edge where the symbol is the output.

Further, there must be a path from the start state to all the states where the transitions labelled with the output symbols originate. This can be verified by applying Flynn‘s algorithm to compute the distance between any two vertices in the graph. Flynn‘s algorithm computes a distance matrix based on an adjacency matrix representation of the graph. Coverage is then ensured if the distance between the start state and all the

output states are greater than zero. Note that with small alphabets the time complexity of computing the distance matrix is insignificant. Also note that this matrix can be reused for several other evaluation measures discussed in subsequent sections.

Error recoverability

Although a text entry design has full coverage it might not be possible to fully recover from errors. We define error recovery as the ability to reach any state in the text entry strategy from any other non-exit state. Typically, a user makes a mistake while selecting the desired letter by walking the graph, and then the error recovery characteristics of the graph will allow the user to reach the desired state without restarting from the origin. In terms of graph theory there should be a path from any state to any output state in the graph for the text entry design to have error recovery capabilities. When using Flynn‘s algorithm (as described in a previous section) all the non-diagonal elements in the distance matrix should be non-zero.

KSPC

KSPC (keystrokes per character) is a measure proposed by MacKenzie [14, 24, 38], which indicates the number of keystrokes required in order to retrieve a particular character. The KSPC measure usually refers to the average KSPC, the minimum KSPC and maximum KSPC. Obviously, KSPC indicates the potential speed in which text can be typed. However, KSPC is criticized in the human computer interaction community for being over simplistic not capturing other important factors affecting typing speed. We do not wish to add to this debate but rather demonstrate how to compute KSPC using the proposed methodology.

Given a graph model of the text interface it is easy to determine minimum maximum and average KSPC. Given a starting state C_{start} and a set of output states $C_{out} \subset C$, then the minimum KSPC is given by:

$$KSPC_{min} = \min_{a \in A} Path(G, C_{start}, C_{out}(a)) \tag{1}$$

The maximum KSPC is given by:

$$KSPC_{max} = \max_{a \in A} Path(G, C_{start}, C_{out}(a)) \tag{2}$$

and finally the average KSPC is given by:

$$\overline{KSPC} = \sum_{a \in A} Path(G, C_{start}, C_{out}(a)) f(a) \tag{3}$$

where $f(a)$ is the probability of occurrence for the character a.

4 Examples of the methodology applied

For the purpose of demonstrating the proposed methodology six text entry strategies are modelled, namely the DateStamp approach (see Figure 1), a new oneway DateStap strategy (see Figure 2), Raghunath and Narayanaswami‘s [30] wristwatch strategy (see Figure 3), multi-tap (see Figure 4), Sandnes et al‘s [35] multi-ring (see Figure 5) and a mesh (see Figure 6). The characteristics of each strategy is summarised in Table 1.

The graph model for the date-stamp method for an alphabet of five characters (“a”, “b”, “c”, “d” and “e”) is presented in Figure 1. It is obvious how this extends to the

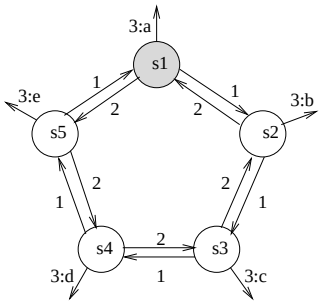

Figure 1: The classic date-stamp text entry strategy

full alphabet (omitted to save space). The user starts in the state indicated with the gray background, and the user can move to the two neighboring states using key 1 or 2. To select the character associated with the state the user presses key 3. Clearly, this design is feasible as each state only has three outbound edges. Further, the design has full coverage, as there is a path from the start state to all the other states. In fact, there is a path from any non-exit state to any other state and the design supports full error correction.

Further, $KSPC_{min} = 1$ as only one transition is needed to produce the output character "a" (assuming snap-to-beginning). $KSPC_{max}$ of this design is identical to the diameter of the graph [18]. The diameter of a graph is defined as the longest path between any two vertices of a graph. For this particular design $KSPC_{max} = 3$. For the English alphabet $KSPC_{max} = 14$. The mean number of keystrokes per character for the English alphabet is KSPC = 7.77.

The graph representation simplifies the understanding of text entry designs, and in this next example shown in Figure 2 a subtle but significant alteration is made to the previous date-stamp approach (This is to the best of our knowledge a yet undocumented strategy). The difference between this strategy and the date stamp approach is that the user only can scroll in one direction, and that the user at each step can select one of two characters.

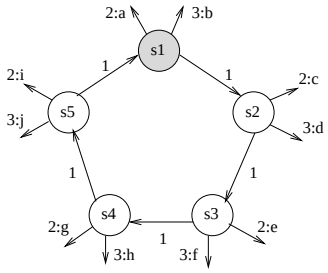

Figure 2: The one-way date-stamp text entry strategy

The example design in Figure 2 also comprises 5 states, but it supports twice as many output symbols (a total of 10 symbols). The design is feasible and has full coverage. Clearly, the $KSPC_{min} = 1$, and $KSPC_{max} = 5$ for this design and $KSPC_{max} = 14$ for the full English alphabet and the mean KSPC = 6.113. This strategy is thus theoretically more efficient than the traditional date-stamp approach. The strategy is error recoverable.

Table 1: Summary of the characteristics for the six text entry strategies.

Test entry strategy	$KSPC_{min}$	$KSPC_{max}$		$\overline{KSPC}$
Datestamp	1	3	14	7.77
Oneway datestamp	1	5	14	6.11
Raghunath et al. wristwatch	1	3	16	7.19
Multitap	1(2)	4	14	6.69
Multi-Ring	2	4	7	4.98
Mesh	1	5	9	4.50

However, the drawback is that the user can only scroll in one direction and the user may have to traverse the entire ring in order to make a correction.

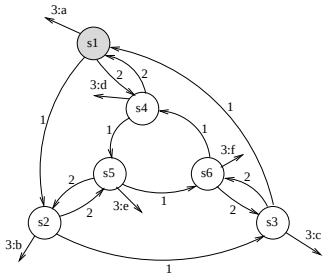

Figure 3: The Raghunath and Narayanaswami wristwatch text entry strategy

The multi-tap approach is a strategy that is widely used on mobile phones. The design in Figure 4 incorporates the characters “a” to “f” where three letters is assigned to each of the two of the keys and the third key is used as a break key. To expand this design to the full alphabet more letters are simply added to each of the multi-tap keys. The design is feasible and has full coverage. However, this strategy is not fully error recoverable. For example if the user starts by pressing key 1 the user is moved to state 2. Imagine that this is a mistake as the user intends to type the character “d”. There is no way go get to state 5 without producing output. If for instance the user chooses to press key 2 to reach state 5 the character “a” is output.

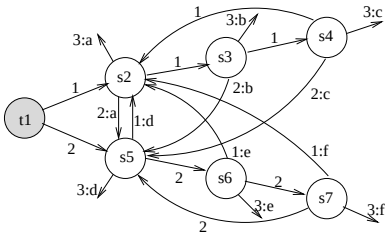

Figure 4: The multi-tap text entry strategy.

Although the multi-ring strategy shown in Figure 5 is correct it is not error recoverable. Once the user enters one of the sub-rings there is no path back. For example,

imagine that the user want to enter the character “d”. The user by accident presses key 3 from the start state 1 which moves the user to state 4. State 4 gives the user access to one of the characters “a”, “b” or “c”. There is no path from state 4 to state 7, which is used to produce the letter “d”.

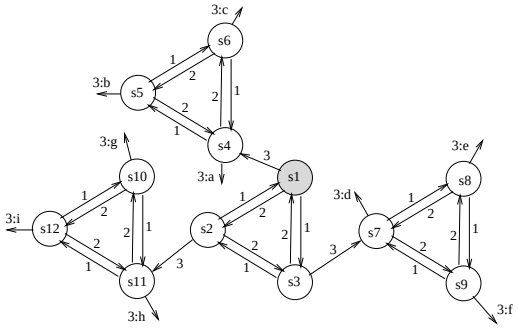

Figure 5: The multi-ring text entry strategy.

The mesh strategy shown in Figure 6 is not previously documented. It is a simplification of the bi-directional wraparound mesh [4] for five- key devices. Three characteristics of the mesh are particularly suitable for mobile text entry: each vertex has few outbound edges (i.e. buttons), the diameter of the mesh structure is relatively small and there is a path between any two states in the structure (error recoverability with a short distance). Figure 6 shows a 3x3 mesh incorporating the characters “a” to “i”. In this design one key is used to cycle vertically, the second to scroll horizontally and the third key is used to select letters. Obviously, the strategy is feasible and has full coverage. Table 1 shows that this strategy is the best when considering the mean KSPC, which is 4.5 (note that these values depends slightly on the character layout). Only the MultiRing has lower $KSPC_{max} = 7$, as opposed to $KSPC_{max} = 9$ for the mesh.

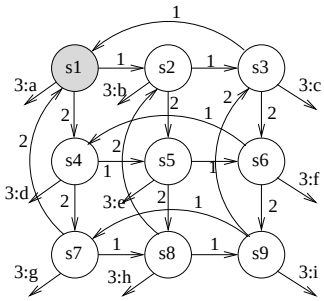

Figure 6: The mesh text entry strategy.

5 Summary

A graph based methodology for the representation, design and evaluation of text entry techniques for miniature mobile devices is presented. The technique allows different text entry strategies to be uniformly described and compared. Further, it is easy to check

for correctness and error recoverability and KSPC. The methodology is not intended as a replacement for typing test using real people on real devices but rather as an early screening tool as it will identify poor text entry strategies early before the interaction engineer commits to expensive testing involving users.

References

[1] S. B. Barnes. Douglas Carl Engelbart: Developing the Underlying Concepts for Contemporary Computing. *IEEE Annals of the History of Computing 19:3*, pp. 16-26, 1997.

[2] M. Beaulieu. *Wireless Internet: Applications and Architecture: Building Professional Wireless Applications Worldwide*. Addison Wesley, 2002.

[3] M. P. Beddoes and Z. Hu. A Chord Stenograph Keyboard: A Possible Solution to the Learning Problem in Stenography. *IEEE Transactions on Systems, Man and Cybernetics 24:7*, pp.. 953-960, 1994.

[4] T. Bellman and I. S. MacKenzie. A Probabilistic Character Layout Strategy for Mobile Text Entry. *In Proc. presented at Graphics Interface '98*, Toronto, Canada, pp. 168-176, 1998.

[5] W. Buxton. A Three-State Model of Graphical Input. *In Proc. presented at Human-Computer Interaction - INTERACT '90*, pp. 449-456, 1990.

[6] W. Buxton and B. A. Myers. A Study in Two-Handed Input. *In Proc. presented at Proceedings of CHI '86*, pp.321-326, 1986.

[7] J. J. Darragh, I. H. Witten, and M. L. James. The Reactive Keyboard: A Predictive Typing Aid. *IEEE Computer 23:11*, pp. 41-49, 1990.

[8] T. Evreinova, G. Evreino, and R. Raisamo. Four-Key Text Entry for Physically Challenged People. *In Proc. presented at 8th ERCIM workshop, User Interfaces 4 All, Adjunct workshop proceedings*, 2004.

[9] P. M. Fitts. The Information Capacity of the Human Motor System in Controlling Amplitude of Movement. *Journal of Experimental Psychology 47*, pp. 381-391, 1954.

[10] M. Fukumoto and Y. Tonomura. Body Coupled Fingering: Wireless Wearable Keyboard. *In Proc. presented at CHI'97*, pp. 147-154, 1997.

[11] D. Goldberg and C. Richardson. Touch Typing with a Stylus. *In Proc. presented at ACM Interchi'93*, pp. 80-87, 1993.

[12] E. Gopher and D. Raij. Typing with a Two-Hand Chord Keyboard: Will the Qwerty Become Obsolete? *IEEE Transactions on Systems, Man and cybernetics 18:4*, pp.. 601-609, 1985.

[13] P. Isokoski and R. Raisamo. Device Independent Text Input: A Rationale and an Example. *In Proc. presented at the Working Conference on Advanced Visual Interfaces AVI2000*, Palermo, Italy, pp. 76-83, 2000.

[14] C. L. James and K. M. Reischel. Text Input for Mobile Devices: Comparing Model Prediction to Actual Performance. *In Proc. presented at CHI'2001*, pp. 365-371, 2001.

[15] M. T. King. Justtype. Tm. - Efficient Communication with Eight Keys. *In Proc. presented at Proceedings of the RESNA 95 Annual conference*, Vancouver, BC, Canada, 1995.

[16] A. Kirchenbaum, Z. Friedman, and A. Melnik. Performance of Disable People on a Chordic Keyboard. *Human Factors 28:2*, pp. 187-194, 1986.

[17] J. G. Kreifeldt. Reduced Keyboard Designs Using Disambiguation. *In Proc. presented at The human factors society 33rd annual meeting*, 1989.

[18] V. Kumar, A. Grama, A. Gupta, and G. Karpypis. *Introduction to Parallel Computing - Design and Analysis of Algorithms*. The Bennjamin/Cummings Publishing Company, Inc, 1984.

[19] J. Lehikoinen and M. Roykee. Lehikoinen, J. And Roykee, M.",. *Interacting with Computers 13*, pp. 601-625, 2001.

[20] S. H. Levine. Multi-Character Key Text Entry Using Computer Disambiguation. *In Proc. presented at RESNA 10th annual conference*, San Jose, California, 1987.

[21] J. Lumsden and A. Gammell. Mobile Note Taking: Investigating the Efficacy of Mobile Text Entry. *In Proc. presented at Mobile HCI 2004*, Glasgow, Scotland, pp. 156-167, 2004.

[22] K. Lyons, D. Plaisted, and T. Starner. Expert Chording Text Entry on the Twiddler One-Handed Keyboard. *In Proc. presented at Eighth International Symposium on Wearable Computers (ISWC'04)*, Arlington, Virginia, pp. 94-101, 2004.

[23] K. Lyons, T. Starner, D. Plaisted, J. Fusia, A. Lyons, A. Drew, and E. W. Looney. Twiddler Typing: One-Handed Chording Text Entry for Mobile Phones. *In Proc. presented at CHI 2004*, pp. 81-88, 2004.

[24] I. S. MacKenzie. Kspc (Keystrokes Per Character) as a Characteristic of Text Entry Techniques. *In Proc. presented at Mobile HCI 2002*, pp. 195-210, 2002.

[25] I. S. MacKenzie. Mobile Text Entry Using Three Keys. *In Proc. presented at NordCHI'02*, pp. 27-34, 2002.

[26] I. S. MacKenzie and R. W. Soukoreff. Text Entry for Mobile Computing: Models and Methods, Theory and Practice. *Human Computer Interaction 17:2*, pp. 147-198, 2002.

[27] A. E. Matias, I. S. MacKenzie, and W. Buxton. Half-Qwerty: Typing with One Hand Using Your Two-Handed Skills. *In Proc. presented at CHI'94*, pp. 51-52, 1994.

[28] J. Moyes. Chord Keyboards. *Applied Ergonomics 14:1*, pp. 55-69, 1983.

[29] K. Partridge, S. Chatterjee, and R. Want. Tilttype: Accelerometer-Supported Text Entry for Very Small Devices. *ACM CHI'01 4:2*, pp. 201-204, 2001.

[30] M. T. Raghunath and C. Narayanaswami. User Interfaces for Applications on a Wrist Watch. *Personal and Ubiquitous Computing 6*, pp. 17-30, 2002.

[31] R. Rosenberg and M. Slater. The Chording Glove: A Glove-Based Text Input Device. *IEEE Transactions on Systems, Man and cybernetics 29:2*, pp. 186-191, 1999.

[32] F. E. Sandnes. "One Handed Text Entry: Evaluation of Five-Key Text Entry Techniques," in IFIP TC8 Working Conference on Mobile Information Systems, E. Lawrence, B. Pernici, and J. Krogstie, Eds. Oslo, Norway: Springer Verlag, pp. 331-339, 2004.

[33] F. E. Sandnes and H.-L. Jian. Pair-Wise Variability Index: Evaluating the Cognitive Difficulty of Using of Mobile Text Entry Systems. *In Proc. presented at Mobie HCI*, Glasgow, Scotland, pp. 347-350, 2004.

[34] F. E. Sandnes, H. W. Thorkildssen, A. Arvei, and J. O. Buverud. Techniques for Fast and Easy Mobile Text-Entry with Three-Keys (Dictionary Based). *In Proc. presented at NIK2003 - Annual National Norwegian Computer Science Conference*, Oslo, Norway, pp. 205-216, 2003.

[35] F. E. Sandnes, H. W. Thorkildssen, A. Arvei, and J. O. Buverud. Techniques for Fast and Easy Mobile Text-Entry with Three-Keys (Non-Dictionary Based). *In Proc. presented at HCISS'37 Hawaiian International Conference on System Science*, Big Island, Hawaii, 2004.

[36] F. E. Sandnes, W. Z. Zheng, and R. M. Shaikh. An Approach to Teaching Interaction Design and Usability Principles: A Mobile Text Entry Perspective. *In Proc. presented at International Conference on Engineering Education and Research ICEER*, Tainan, Taiwan, 2005.

[37] R. Seibel. Performance on a Five-Finger Chord Keyboard. *Journal of Applied Psychology 46:3*, pp. 165-169, 1962.

[38] M. Silfverberg, I. S. MacKenzie, and P. Korhonen. Predicting Text Entry Speed on Mobile Phones. *ACM CHI'2000 1:1*, pp. 9-16, 2000.

[39] K. Tamaka-Ishii, Y. Inutsuka, and M. Takeichi. Entering Text with a Four-Button Device. *In Proc. presented at The 19th International Conference on Computational Linguistics COLING 2002*, Taipei, Taiwan, pp. 988-994, 2002.