Effective Virtual Keyboard Design with Size and Space Adaptation

Soumalya Ghosh[1], Sayan Sarcar[2], Manoj Kumar Sharma[3] Debasis Samanta[4]
IIT Kharagpur, Kharagpur - 721302, West Bengal
Email: [1]sghosh@sit.iitkgp.ernet.in, [2]sayans@sit.iitkgp.ernet.in, [3]manojs@sit.iitkgp.ernet.in [4]dsamanta@sit.iitkgp.ernet.in

***Abstract*—In recent years, there are huge developments in Information and Communication Technology (ICT). ICT in present form has offered to interact with the computing system in users own language. But, there are several challenges to be addressed to achieve the full advantages of ICT. One of the biggest issue is standard text input mechanism. In practice, the text input mechanism is mainly realized through the virtual keyboard. A numbers of work have been proposed to design virtual keyboard with optimum layout. So that performance of text entry can be enhanced. Still there are demand of more improvement to make an efficient text entry mechanism. In this paper, a new concept of adaptation in virtual keyboard is proposed. Our objective is to develop a user friendly interface through adaptation of key size and centroid distance between keys. As a mean of adaptation, we provide several states (keyboard) on the basis of key size and center distance between keys. Upper and lower threshold values for a performance metric of a state are defined by evaluating the states using motor movement time and searching time. If one individual user performance is in between the range of upper and lower threshold values for the keyboard then no adaptation is required. Otherwise, the keyboard is adapted according an algorithm proposed in this paper. The proposed approach is experimented with real users and results substantiate the efficacy of the approach.**

***Index Terms*—Virtual keyboard, Fitts-Digraph model, keyboard adaptation and personalization, Information and communication technology, human computer interaction.**

I. Introduction

Virtual keyboard [1] [2] [3] is one of most primitive alternative text entry mechanism [4] [3] for entering text. It replaces the hardware keyboard through on screen character image map. It is a challenging task to arrange large number of keys of hardware keyboard into small hand held device interfaces. Special helping mechanisms like word completion [5], word prediction [6], adaptation and personalization [7], can also be easily incorporated with virtual keyboard to increase the user performance, which are found to be difficult to integrate into hardware keyboard. These above mentioned scenarios are the cause of the virtual keyboard design as a explorable field in present days to reduce gap between man and machine.

To achieve efficient text entry speed using virtual keyboard, various methods and tricks [6] [5] [7] have been applied. They are namely keyboard layout and key size optimization [3], next word or character prediction [8], typing error detection and correction [9], personalization and adaptation etc. In this paper, we address virtual keyboard adaptation issues for achieving more user friendliness and efficiency.

According to definition, *'adaptation'* is one of the basic phenomena of biology [10]. *'Adaptation'* is an evolutionary process through which an organism becomes more able to live in its habitat or habitats. This biological phenomenon is also applied to the system for better comfort of user. The concept *"adaptation of soft keyboard"* can be deployed through an keyboard interface of a system. The entire mechanism consists of several phases. The system initially monitors user performance at the time of using the interface. After certain time, the interface comes up with improved organizational design of the keyboard which increases the users' comfort. There are mainly two types of adaptation on keyboard, namely keyboard key layout adaptation and size adaptation. In layout adaptation, the interface periodically measures user performance on virtual keyboard attached with the interface and compares the user performance with the standard performance of virtual keyboard. If the comparison result does not satisfactory, it changes the key position to provide better user satisfaction. Major problem with this approach is if n number of keys exist into the keyboard, then the number of possible layout is $n!$. Hence, choosing optimal design from the large number of designs is quite difficult. Furthermore, frequent changing of layout increases the users' key searching time, which is a parameter to increasing typing speed and user becomes confused with new layout. in every time. In this concept, system does provide a very little time (one state) to familiar with the key layout design of keyboard. In size adaptation, system periodically measures user performance, compare with the standard performance of virtual keyboard and then change the key size to provide better user satisfaction. Main concept is to periodically vary the size to achieve the optimal motor movement and typing speed for any user. In this concept, as the relative position of key and key arrangement does not change, so the searching time in the keyboard does not change and also user does not confuse as the layout relative key position and arrangement remain same in every time.

The paper is organized as follows. Section II states about existing work related with adaptive keyboard design issues. In section III, we describe the basic methodology. We also precisely elaborate the implementation methodology in section IV. User testing is discussed in section V. Finally, section VI concludes the paper with future work discussion.

 978-1-4244-5974-2/10/$26.00 © 2010 IEEE

II. State of the Art

Several work have been done in designing of effective virtual keyboard. Effectiveness is concern towards less motor movement, increasing typing speed [11]. Work is done to reduce the physical motor movement by changing the key arrangement in keyboard. *Chubon* keyboard layout [12] use the same approach as in the case of the Qwerty keyboard layout [13]. However this layout is designed for ten fingers typing and inefficient for stylus based single pointer typing.

MacKenzie and Zhang[3] design a new layout, called *'dubbed Opti'*. They first place the ten most frequent letters in the center of the keyboard. Then they assign the top ten key corresponding to most frequent in digraph [14]. Finally, they place the remaining letters in keyboards. The placement is done by trial and error method. Further improvement is made in 5×6 layout.

The *Fitaly* keyboard [15] use the idea of layout design which includes center placement of more frequent keys, dual double sized space key and the consideration of digraph frequencies.

"Dynamic simulation method" [16] in spring simulation for the design of virtual keyboard is developed by S.Zhai, M.Hunter and B.Smith [1]. To construct the initial string, the keys are placed randomly with spaces between them. The elasticity of each spring is proportional to the digraph probability between the two keys and also the keys with higher digraph probability are pulled together with greater forces. In addition, viscous friction was assumed to be present between the keys and their environment. All keys are pulled together to form a candidate design in steady state. The movement efficiency of the design is then calculated according to the Fitts-Digraph model [3] [2]. When unsatisfactory results come, the layout could be "stretched" out to serve as another initial state and iteratively do the same process. The iterations are repeatedly done until a satisfactory state is come."Metropolis" [1] algorithm is developed on the base of "Fitts-Digraph energy" [1] which also derived from FD model.

M.Raynal and N.Vigouroux[17] proposes a mechanism to solve the keyboard optimization problem using the concept of genetic algorithm. In this approach, each layout is treated as an individual entity. The initialization of the first population is done by placing the key randomly. The individuals can live, reproduce and die according to the fitness function. The process is executed iteratively until an optimal layout has come up. All the above methods are used to develop a better key arrangement towards reducing the physical motor movement. As a result, typing speed must be increased.

All the work have been done for general user. The users who are unable to cope with the general design, necessary changes are required to enhance individual user performance. Unfortunately, no standard work has been reported in this direction till date. In this paper, we address this problem.

III. Basic Methodology

User typing performance of a keyboard is a function of text entry speed and typing error rate. It can be stated in either way that how fast a user can type with very less or minimum error. User typing speed is directly related to *"Mean Motor movement time" (MT_{MEAN})* and *Visual search time or Reaction time (RT)* in the time of typing. User typing speed is calculated according to eqn 1. Mackenzie et al.[3] [2] are the first to use a quantitative approach for modelling virtual keyboard performance. Their model predicts users' performance by summing the Fitts' law [11] movement time between all digraphs, weighed by the frequency of occurrence of the digraphs. Applying Hick-Hyman law *Reaction time or Visual search time* is then calculated [18] [19].

$$T = \sum_{i=0}^{n} \sum_{j=0}^{n} (MT_{ij} \times P_{ij}) + RT \tag{1}$$

Movement time from i^{th} key to j^{th} key with in keyboard is depicted in Eqn. (2).

$$MT_{ij} = a + b \log_2 (D_{ij}/W_j + 1) \tag{2}$$

Where a and b are empirically determined coefficients and d_{ij}, w_j signify distances between i and j^{th} key and key size accordingly.
The probability of typing of j after i^{th} key is shown in Eqn. (3).

$$P_{ij} = f_{ij} / \sum_{i=1}^{N} \sum_{j=1}^{N} f_{ij} \tag{3}$$

Where n represents total number of characters present in the keyboard. Reaction time for typing in Hick-Hyman law is given in Eqn. (4).

$$RT = a^{'} + b^{'} \log_2 N \tag{4}$$

Where N is the number of keys in the keyboard and $a^{'}$ and $b^{'}$ are constants and value are determined by empirically.

In our developed system, particular user performance is determined by calculating internal log file of the system. If user performance is not matched with mathematically calculated standard performance, then the system is automatically adapting other keyboard layout which performs towards the user performance to give more comfort and satisfaction to targeted user.

IV. Proposed Methodology

In this section, we are going to discuss our proposed idea about key size adaptation on the basis of previously described basic methodology. After analyzing the basic formulae, it has been observed that typing speed is directly related with key width (w_i) and center distance between two keys (d_{ij}). It is clear that movement time is inversely proportional to the key width and directly proportional to space between them only when key location should be equally shifted through x axis.

But according to Eqn. (2), fixed location of the keys and the distance must be fixed.

$$\begin{aligned} &Movement\ Time = f(key\ size, distance\ between\ key) \\ &Movement\ Time \propto 1/w_i \\ &Movement\ Time \propto d_{ij} \end{aligned} \tag{5}$$

We take a set of different keyboards with varying key size and distance between the keys. We calculate the movement time (MT) for each keyboard and represent in matrix form as shown in Eqn. (6).

$$S_{keyboard} = \begin{bmatrix} MT_{d_1w_1} & MT_{d_1w_2} & \cdots & MT_{d_1w_n} \\ MT_{d_2w_1} & MT_{d_2w_2} & \cdots & MT_{d_2w_n} \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ MT_{d_mw_1} & MT_{d_mw_2} & \cdots & MT_{d_mw_n} \end{bmatrix} \quad (6)$$

In Eqn. (6), each element of the matrix represents the movement time of the keyboard with key width of w_j and distance between two keys of d_i. Each row of the matrix consists of a movement time of fixed distance but varying key width ($w_1 < w_2 < .. < w_n$). The elements in each row are constituted in descending fashion means, the movement times got minimized for nth column of the element for each row ($MT_{d_iw_n}$) and maximized for first element ($MT_{d_iw_1}$). For each column, the width of keys are fixed but distance are varying ($d_1 < d_2 < .. < d_m$). The MT distribution in a row follows constraints that the minimum valued MT for each row is greater than every element of the previous row ($MT_{d_{i-1}w_j}$) and value of $MT_(max)$ for the row is smaller than every MT value of the next row ($MT_{d_{i+1}w_j}$). The situation is depicted in Eqn. (8).

$$MT_{d_iw_j} > MT_{d_iw_{j+1}} \quad (7)$$

$$for\ 1 \le i \le m \quad and\ 1 \le j \le n-1$$

$$MT_{d_iw_1} < MT_{d_{i+1}w_n} \quad (8)$$

Where, $1 \le i \le m-1$

We add and subtract a very small percent of MT value with each Movement Time ($MT_{d_iw_j}$) in $S_{keyboard}$ for real user individual performance which are called upper threshold $MT_{d_iw_j}TH_u$ and lower threshold value $MT_{d_iw_j}TH_l$ of *Movement Time* of the keyboard. Following the rule, we create a *Movement Time* upper and lower range for all keyboards. The scenario can be mathematically explained in Eqn. (9) & (10) for better understanding.

$$MT_{d_iw_j}TH_u = MT_{d_iw_j} + x\%MT_{d_iw_j} \quad (9)$$

$$MT_{d_iw_j}TH_l = MT_{d_iw_j} - x\%MT_{d_iw_j} \quad (10)$$

Where, $1 \le i \le m$ and $1 \le j \le n$

If user *Movement Time* of a keyboard lies between $MT_{d_iw_j}TH_u$ and $MT_{d_iw_j}TH_l$ then that keyboard is suitable for a particular user. Otherwise, the keyboard is not decided as a suitable for user and system needs to be changed itself towards the user comfort.

Row Movement Time is calculated by summing up the all elements in a row of the matrix $S_{keyboard}$ as shown in Eqn. (11).

$$MT_r = \sum_{j=1}^{n} MT_{d_rw_j} \quad (11)$$

Where, $1 \le r \le m$ and $1 \le j \le n$

Upper (lower) threshold values for every row movement time is calculated by adding (subtracting) very small percent of row movement time. We use these threshold values to decide the targeted keyboard with user comfort quickly. The upper and lower threshold for each row movement time is defined in Eqn. (12) & (13). respectively.

$$MT_rTH_u = MT_r + x\%MT_r \quad (12)$$

$$MT_rTH_l = MT_r - x\%MT_r \quad (13)$$

Where, $1 \le r \le m$

At this point we see that total number of keyboards available is number of entries present in $S_{keyboard}$ that is $m \times n$. If we search the key in sequential manner then maximum run time needed is equal to number of elements present. We define special two way binary search tree data structure to optimize the search time of the keys in keyboard significantly. This above mention data structure is basically a special case of binary search tree. Every node is two ways except the root node. Before going to discuss our methodology in detail. We have defined special two way binary search tree data structure.

For example, $T_1, T_2, T_3 \cdots T_n$ are n number of binary search trees and all key values of T_2 are greater than all key values of T_1 and less than T_3 (i.e. $T_1 < T_2 < T_3$). If we want to merge all T_1, T_2, T_3 into an empty special two way binary search tree T, then the root node of T_2 is the root node of T and also root node of T_1 and T_3 is appended with left most and right most node of T_2 correspondingly as shown in Fig. 1. We then add two data fields, namely the highest key value and the lowest key value of that sub tree, in every root node of individual sub trees to reduce the search time. At the time of searching, if the search item value belongs to intermediate of those key values then only search entire sub tree and if it is not find in the sub tree, than item is not present in the tree. When the search item does not belong to the sub tree, then search proceeds to the predecessor or successor of sub tree maintaining the property of binary search tree. Furthermore, we have to modify link field of each root node in sub tree by adding two more link field for storeing the address of predecessor or successor of sub tree root providing direct connection among them. We consider total number of sub trees is m and total number of node of each sub tree is n. So, special two way binary search tree contains $m \times n$ number of node. Our proposed special two way binary search tree is a special case of binary search tree. We compare our proposed special two way binary search tree with simple binary search tree in respect to search time.

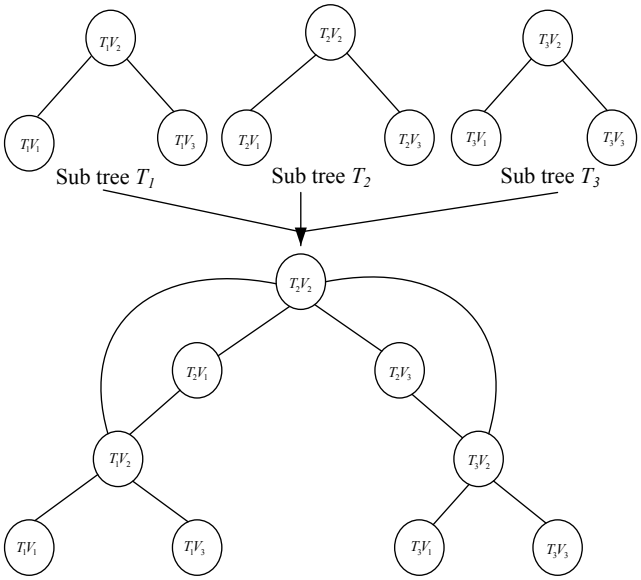
Binary search tree = $log_2\, mn$

Special two way binary search tree = $log_2\,(m \times log_2\, n)$

So, search time in our special two way binary search tree is less than binary search tree.

Algorithm 1 will illustrate the method for searching desired data value on proposed special two way binary search tree.

To find optimality of individual user appropriate keyboard, we apply our proposed special two way binary search tree data structure on $S_{keyboard}$. We design a binary search sub tree for each row in $S_{keyboard}$. Then we merge all of those binary search trees and create a special two way binary search tree. Then we apply special two way binary search tree searching



LC	VAL	RC	LV	UV	PR	SR

Structure of Root node of Sub Tree

LC	VAL	RC

Structure of a node

LC- Left child
RC-Right child
VAL-value of the node
LV-Lowest value of the sub tree
UV- Upper value of the sub tree
PR-Address of predecessor sub tree root node
SR- Address of successor sub tree root node

Fig. 1. Special two way binary search tree

algorithm to search the appropriate keyboard on the basis of user typing speed and pre calculated typing speed for the keyboard. In the present context, the development keyboards with size and distance between the keys specification are applicable in desktop based computers where screen size is moderately large. In contrast, the robust algorithm can be fitted with several small size and distance values in any small handheld devices like mobile phone, iPad, Tablet PC, palmtop etc.

V. System Implementation

To develop the entire system, we use .Net 3.5 framework in windows environment. First, we design total 21 virtual keyboards in Bengali language with different key size and distance between two adjacent keys. We choose the key size and distance parameter of all keyboards in such a way that it satisfies the need of $S_{keyboard}$ matrix of Eqn.(6). All the specifications are shown in the table I. All keyboards are arranged in alphabetical order of Bengali language. We have considered all the vowels, consonants and matras of Bengali language that is 62 characters to calculate Mean Movement Time for each i^{th} key to j^{th} key using the equation of T in Eqn.(1) as shown in the table I.

Then, using mean movement time, average typing time for 3001 character text is measured and also we consider .0005 unit as threshold for every keyboard. Calculated upper and lower threshold values are enlisted in the table I using the Eqn.9 and Eqn.10, correspondingly. Then we create three sub trees, namely T_1, T_2, T_3. Where T_1 contains the node value of $K_{d_1w_1}$ to $K_{d_1w_7}$ and T_2, T_3 contain the node value of $K_{d_2w_1}$ to $K_{d_2w_7}$ and $K_{d_3w_1}$ to $K_{d_3w_7}$, respectively. Each node of a sub tree has a lower and upper threshold value as mentioned in above table I. Then we merge all sub trees into a special two way binary search tree as illustrated in Fig. 1. Lower and upper value of each sub tree is calculated and maintained in table I. When a user first time uses the system, the keyboard of root node ($K_{d_2w_4}$) is given. After

Algorithm-1 *searching in special two way binary search tree*

Input : ITEM is the data that has to be searched
Output: If found then pointer to the node containing as data ITEM else a message.

ptr =ROOT, flag = False, flag_sub = False
while *(ptr* $\neq$ *Null) and (flag_sub=False)* **do**
  case *1*
  | ITEM $< ptr \rightarrow LV$
  end
  $ptr = ptr \rightarrow PR$
  case *2*
  | ITEM $> ptr \rightarrow LV$
  end
  $ptr = ptr \rightarrow SR$
  case *3*
  | $ptr \rightarrow LV < ITEM < ptr \rightarrow UV$
  end
  $flag\_sub = TRUE$
end
while *(ptr*$\neq$*Null) and (flag_sub = TRUE) and (flag = False)* **do**
  case *1*
  | $ITEM < ptr \rightarrow VAL$
  end
  $ptr = ptr \rightarrow LC$
  case *2*
  | $ptr \rightarrow VAL = ITEM$
  end
  $flag\_sub = TRUE$
  case *3*
  | $ITEM > ptr \rightarrow VAL$
  end
  $ptr = ptr \rightarrow RC$
end
if *(flag_sub = TRUE) and (flag = TRUE)* **then**
  | *Print "ITEM has found at the node", ptr*
end
else
  | *Print "ITEM does not exist: Search is unsuccessful"*
end
Stop

TABLE I
ACTUAL MOVEMENT TIMES

Sub Tree	Keyboard	Distance (in cm)	Width (in cm)	Calculated MT_{MEAN}	Typing Time 3001 characters	Upper Threshold (sec)	Lower Threshold (sec)	Lower value of the Sub Tree	Upper value of the Sub Tree
T_1	$K_{d_1w_1}$	1.1	0.94	2.234169	6702.50616	6706	6699	6618	6706
	$K_{d_1w_2}$	1.1	0.96	2.229403	6688.20784	6691	6685		
	$K_{d_1w_3}$	1.1	0.98	2.224753	6674.25831	6677	6671		
	$K_{d_1w_4}$	1.1	1	2.220214	6660.64300	6664	6657		
	$K_{d_1w_5}$	1.1	1.02	2.215783	6647.34824	6651	6644		
	$K_{d_1w_6}$	1.1	1.04	2.221454	6634.36113	6638	6631		
	$K_{d_1w_7}$	1.1	1.06	2.211454	6621.66955	6625	6618		
T_2	$K_{d_2w_1}$	1.3	0.94	2.272618	6817.85481	6821	6814	6731	6821
	$K_{d_2w_2}$	1.3	0.96	2.267713	6803.13848	6806	6800		
	$K_{d_2w_3}$	1.3	0.98	2.262925	6788.77444	6792	6785		
	$K_{d_2w_4}$	1.3	1	2.258249	6774.74805	6778	6771		
	$K_{d_2w_5}$	1.3	1.02	2.253682	6761.04560	6764	6758		
	$K_{d_2w_6}$	1.3	1.04	2.249218	6747.65415	6751	6744		
	$K_{d_2w_7}$	1.3	1.06	2.244854	6734.5615	6738	6731		
T_3	$K_{d_3w_1}$	1.5	0.94	2.306389	6919.16641	6923	6916	6830	6923
	$K_{d_3w_2}$	1.5	0.96	2.301375	6904.12493	6907	6901		
	$K_{d_3w_3}$	1.5	0.98	2.296479	6889.43816	6893	6886		
	$K_{d_3w_4}$	1.5	1	2.291697	6875.09146	6878	6878		
	$K_{d_3w_5}$	1.5	1.02	2.287024	6861.07106	6864	6858		
	$K_{d_3w_6}$	1.5	1.04	2.282455	6847.36400	6851	6844		
	$K_{d_3w_7}$	1.5	1.06	2.277986	6833.95805	6837	6830		

measuring the performance of the user from system log file, suitable keyboard for that user has been selected and swapped with the previous from special two way binary search tree. The most suitable keyboard for that user is identified by repeating the same several times.

VI. USER TESTING

To validate the theoretical analysis in a practical way, we have to perform the user evaluation for the system. We have chosen six user, four male and two female of under graduate and post graduate students, having different specialization from our institute.

Before the actual testing, we have conducted some introductory training, i.e. two hour typing practice to familiarize the user with the key arrangement of the system. Testing phases are conducted in our laboratory equipped with computers having 54cm touch screen monitors. We have selected 50 different text corpuses, containing equal number and same type of characters, for testing. Among these a text corpus has chosen randomly and given to a user for typing. After one set typing is over 15 minute interval has been provided to the user. All testing result and corresponding decision by the system are enlisted in table II. Table II signifies that when a user does not follow calculated typing time, a standard typing speed keyboard near to the system calculated time is provided to the user. The unsuitability of system with user can be decided after certain iterations, if the resultant keyboard is still unable satisfy the user.

TABLE II
USER PERFORMANCE VALUE

User	User Typing time(in sec.)	Standard typing time(in sec.)	keyboard	Decision
User1	6786	6774.7	$K_{d_2w_4}$	$K_{d_2w_3}$ keyboard is suitable for user1
	6790	6788.7	$K_{d_2w_3}$	
User2	6652	6774.7	$K_{d_2w_4}$	$K_{d_1w_6}$ keyboard is suitable for user1
	6635	6647.3	$K_{d_1w_5}$	
	6632	6634.3	$K_{d_1w_6}$	
User3	6764	6774.7	$K_{d_2w_4}$	$K_{d_2w_5}$ keyboard is suitable for user3
	6762	6761	$K_{d_2w_5}$	
User4	6849	6774.7	$K_{d_2w_4}$	$K_{d_3w_3}$ keyboard is suitable for user4
	6875	6847.3	$K_{d_3w_6}$	
	6890	6875	$K_{d_3w_4}$	
	6892	6889.4	$K_{d_3w_3}$	
User5	6915	6774.7	$K_{d_2w_4}$	No keyboard is suitable for user5
	6898	6833.9	$K_{d_3w_7}$	
	6892	6833.9	$K_{d_3w_7}$	

VII. CONCLUSION AND FUTURE WORK

In this system we are concerned about how the typing time varies with the size of keys and distance among them. We have calculated the theoretical typing time for each set of text in each keyboard to compute user evaluation performance. When the user performance does not match with standard performance, the system automatically adopts with the user by modifying size and distance of keys to provide more suitable keyboard for enhancing user performance.

In this paper we address only the able bodied general user are the appropriate user for our system. In near future we classified the user in several groups in the basis of their physical fitness, educational and social background and define a set for each group and then finally adaptation for each group is applied. Some personalization for the needs of individuals is also makes the system more useful for the user. Adaptation through visual effects and analyzing the previous typed text can also be introduced with the system to more optimize the user performance.

REFERENCES

[1] S. Zhai, M. Hunter, and B. A. Smith, "The metropolis keyboard - an exploration of quantitative techniques for virtual keyboard design," in *Proceedings of the ACM Symposium on User Interface Software and Technology - UIST*. New York: ACM, 2000, pp. 119–128.

[2] R. W. Soukoreff and I. S. MacKenzie, "Theoretical upper and lower bounds on typing speed using a stylus and soft keyboard," *Behaviour and Information Technology*, vol. 14, pp. 370–379, 1995.

[3] I. S. MacKenzie and S. X. Zhang, "The design and evaluation of a high-performance soft keyboard," in *Proceedings of the CHI 99 Conference on Human Factors in Computing Systems*, 1999.

[4] D. J.Ward, A. F. Blackwell, and D. J.C.MacKay, "Dashera data entry interface using continuous gestures and language models," in *Proceedings of the 13th annual ACM symposium on User interface software and technology*. ACM New York,NY,USA, 2000, pp. 129–137.
[5] J. O. Wobbrock and B. A. Myers, "From letters to words: Efficient stroke-based word completion for trackball text entry," in *In Proceedings of the ACM SIGACCESS Conference on Computers and Accessibility (ASSETS 06)*, October 2225 2006, p. 29.
[6] C. Aliprandi, N. Carmignani, and P. Mancarella, "An inflected-sensitive letter and word prediction system," *International Journal of Computing & Information Sciences*, vol. 5, no. 2, pp. 79 – 85, 2007.
[7] J. Himberg, J. H. P. Kangas, and J. Mäntyjärvi, "On-line personalization of a touch screen based keyboard," in *Proceedings of the 8th international conference on Intelligent user interfaces*. ACM New York, NY, USA, 2003, pp. 77–84.
[8] K. Trnka, D. Yarrington, J. McCaw, K. F. McCoy, and C. Pennington, "The effects of word prediction on communication rate for aac," in *NAACL '07: Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers on XX*, vol. Companion Volume. Morristown, NJ, USA: Association for Computational Linguistics, Rochester, NY, April 2007, p. 173176.
[9] K. Kukich, "Techniques for automatically correcting words in text," in *ACM Computing Surveys*. ACM New York, NY, USA, 1992, pp. 377–439.
[10] "http://en.wikipedia.org/wiki/evolution."
[11] P. M. Fitts, "The information capacity of the human motor system in controlling the amplitude of movement," *Journal of Experimental Psychology*, vol. 47, p. 381391, 1954.
[12] "http://www.orin.com/access/softype/layouts.htm,the chubon keyboard. 1999."
[13] "Qwerty - wikipedia," http://en.wikipedia.org/wiki/QWERTY (Accessed on January 2010).
[14] "http://en.wikipedia.org/wiki/bigram."
[15] "http://fitaly.com/fitaly/fitaly.htm."
[16] D. Baraff, "Analytical methods for dynamic simulation of non-penetrating rigid bodies," in *Proceedings of the 1989 ACM SIGGRAPH conference*, 1989, pp. 223 – 232.
[17] M.Raynal and N.Vigouroux, "Genetic algorithm to generate optimized soft keyboard," in *CHI 2005*, Portland, Oregon, USA, 2005.
[18] W. E. Hick, "On the rate of gain of information," *Quarterly Journal of Experimental Psychology*, vol. 4, pp. 11–26, 1952.
[19] R. Hyman, "Stimulus information as a determinant of reaction time," *Journal of Experimental Psychology*, vol. 45, pp. 188–196, 1953.