2009 Fifth International Conference on Autonomic and Autonomous Systems

# Measuring Performance of Virtual Keyboards Based on Cyclic Scanning

A.J. Molina, O. Rivera, I. Gómez
Department of Electronic Technology
University of Seville
Seville, SPAIN
almolina@us.es, octavio@dte.us.es, igomez@us.es

***Abstract*—This paper presents an exhaustive study into the different topologies of virtual ambiguous keyboards that operate by scanning techniques, analyzing the text entry average time ($t_c$) and the average number of user inputs ($UI_c$) per character.**

**An mathematical model shows that in comparison with unambiguous one, text entry, in multi-tap mode, doesn't offers better performance,because both $t_c$ and $UI_c$ are greater in them. Another method of text entry, called Tnk (Text in n keys), offers improvement with respect to unambiguous keyboards. But solely highly ambiguous keyboard (4-keys keyboards) shows a jointly reduction in $t_c$ and $UI_c$ . Results obtained with the model do to focus on highly ambiguous keyboard. This paper demonstrate, using simulation with extensive text, that character prediction with TnK mode only have better performance than unambiguous keyboard with character prediction in $UI_c$ parameter. Another techniques of text entry are also studied.**

## I. Introduction

There exists various techniques that allow handicapped individuals to write text; increasing text entry rate and/or reducing the number of user inputs. One of them is called on screen or virtual keyboard (VK). A VK is a application software that paints a keyboard layout on the computer screen. A typical comparison amongst the different keyboards consists in obtaining some parameters simulating keyboard behavior in a computer. A extensive text obtained from sources such as digital journals, magazines, etc, could be used as input of the simulated keyboard [1,2]. Simulations determine estimations about text entry rate and number of user inputs per character

Virtual keyboards (VK) are usually set in a rectangular matrix of keys, each one of them, could contain a different amount of characters in them (although optimal VK's can have a button arrangement different than the rectangular matrix [3], the use of rectangular ones is very common [2,4,5,6]. A VK that has one character in each key is called an unambiguous keyboard. On the other hand, an ambiguous VK contains more than one character on one or more keys. These ones require the disambiguation of the character contained in the key.

Access to the keys of the VK are performed by cyclic scanning. In linear scan, each key is highlighted, one after the other. Linear scanning is useful when the keyboards are highly ambiguous (no more than 4 keys). This is not the case in other keyboards where it is best to perform row-column scans. In them, each row of matrix is highlighted, one after other. A user input, when a row is highlighted, triggers a row selection and the beginning of linear scan over the keys contained in the pre-selected row, other scanning method are not considered in this work. The scan can be automatic or manual. In the first, a timer establishes when the row, key or character is highlighted. In the latter, the user input triggers the advance of highlighting. This paper focuses mainly on the automated scans, because only requires one user input: selection.

## II. Chosen Topologies

In this paper fixed character layout are chosen. Character distribution in them is based on frequency in Spanish language. These keyboard are considered optimal because tex input rate ($t_c$) is increased without affects of number of user input ($UI_c$) like it is mentioned in the introduction .

Figure 1 shows an optimal unambiguous keyboard that is the reference for this analysis, this is the conventional keyboard (called CONV). It can be observed that punctuation marks and accented vowels are not considered in this study.

Figures 2,3,4,5,6 show ambiguous keyboard analyzed (called REDn where n is the number of keys). The layouts choice allows to study how affects to the performance the grade of ambiguity.

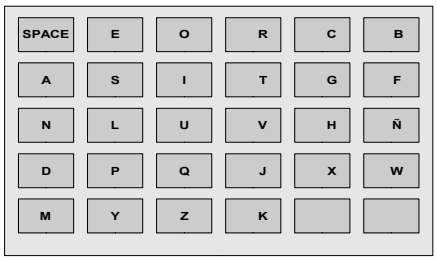


Figure1. Optimal unambiguous keyboard for Spanish.

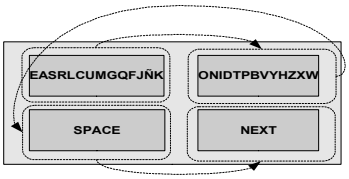


Figure 2.Ambiguos keyboard with four keys (called RED4).

978-0-7695-3584-5/09 $25.00 © 2009 IEEE
DOI 10.1109/ICAS.2009.35

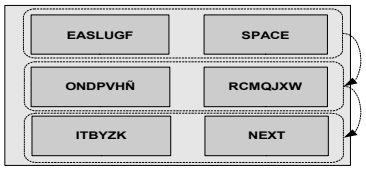


Figure 3. Ambiguos keyboard with six keys (called RED6)

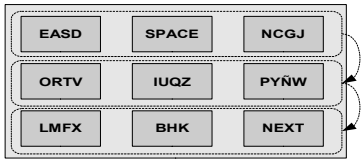


Figure 4. Ambiguos keyboard with nine keys (called RED9)

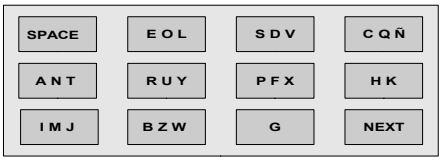


Figure 5 Ambiguos keyboard with twelve keys (called RED12)

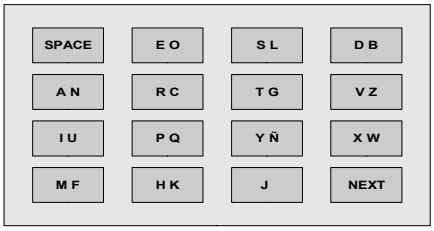


Figure 6. Ambiguous keyboard with sixteen keys (called RED16)

In all the ambiguous layouts a NEXT key is introduced. This key is necessary for one of a the text input method explained in section 4 (called Tnk). The other text input method have same layout but does not have NEXT key and character distribution is not the same.

## III. Mathematical Model

The average time used to access a character $t_c$ is given by relationship 1, where p represents a probabilities matrix of character in a alphabet and $t_l$ the access times matrix of VK.

$$\bar{t}_c = p^T . t_l \qquad (1)$$

Assuming that word average length, including space character, is 5.5 for Spanish, equation 2 gives words per minute (WPM).

$$WPM=60/(5.5\ t_c) \quad (2)$$

The UI average is obtained the same way. Let $ui_l$ be the matrix that represents the numbers of necessary user inputs to access a specific character in VK. Then, $UI_c$ is given by equation 3.

$$\bar{UI}_c = p^T . ui_l \qquad (3)$$

A parameter that affects the WPM is the time scan $T_{scan}$, which is included in the matrix $t_l$. If the scan is done quickly the number of written words is increased. Studies [4,5,8] have shown that the optimal time scan is related to the reaction time of the user (tr) by a constant equal to .0.65: $T_{scan} \geq tr/0.65$. The scan time appears in all of the components of the matrix $t_l$: the row scan, the column scan and letters scan within each key. All of them are performed in a time equal to $T_{scan}$ . To assign a value to $T_{scan}$ in itself is not relevant to the effects of establishing comparisons of methods and VKs for this reason, we will set this parameter to one. The calculation of $t_c$ for other $T_{scan}$ can be gotten by multiplying the result obtained from the unitary reference by the scanning time.

## IV. Text Input Methods

In this section mathematical model is applied to the topologies under test using two text inputs methods: multitap and TnK(text in n keys).

### *A. Ambiguous keyboard with multi-tap operation*

Access to the letters on an ambiguous keyboard initially requires access to the key and, and then, access to the character it contains. Table 1 contains $t_c$, $UI_c$ and WPM for different keyboards that use row and column scans to access the virtual keys, and linear for the selection of the letter within the key (REDnkRC where nk is the number of keys and RC represents row-column scanning). The corresponding parameters of a four keys VK with a lineal scan are also represented (RED4L).

It can be seen that $t_c$ progressively decreases in relation to the increase of ambiguity (with exception of RED4RC keyboard because it uses an unappropriated scanning method), reaching a minimum in VK with 6 keys. In all of them WPM is always less than an unambiguous one(CONV), and they show a greater $UI_c$. A keyboard with only 4 keys, when using a lineal scan (RED4L), is able to get a $t_c$ close to the unambiguous one and with an identical $UI_c$.

TABLE I Multi-tap Operation Mode Results for Different Keyboard Layouts Using Row-Column or Linear Key Scanning, Linear Scanning for Character into the Key and Tscan = 1s.

| layout | tc(s) | UIc | WPM |
|---|---|---|---|
| red4l | 4.23 | 2.00 | 2.58 |
| red4rc | 4.91 | 3.00 | 2.22 |
| red6rc | 4.56 | 3.00 | 2.39 |
| red9rc | 4.63 | 3.00 | 2.36 |
| red12rc | 4.63 | 3.00 | 2.36 |
| red16rc | 4.70 | 3.00 | 2.32 |
| conv | 4.18 | 2.00 | 2.61 |

*B. Ambiguous keyboards with Tnk operation mode*

There exists a method of text entry on mobiles, called T9 (text in 9 keys), property of Tegic Communication, that accelerates the process of writing when it is compared to the multi-tap method. In [9] it is shown that WPM is bigger in T9 than using multi-tap, and in [10] a comparative table for mobile phone can be seen, where the KSPC (keystrokes per character), with a T9 method, is close to one.

The Tnk method (generalization of T9 for nt keys) requires a dictionary and a search engine. Initially, the user selects keys that contain characters of the words that the user wants to write. As the writing continues, the system shows the most likely word associated to the selected keys sequence. In most cases (95% according to [9] for mobile phones), the system presents the word sought. So, the user only needs to select the SPACE key to accept the suggested word and continue writing. Only in 5% of the cases the suggested word is not accepted. Under these circumstances, the user should use the NEXT key in order to see the other options. Also, the word sought may not be found in the dictionary and obviously will not be shown as an option. This is the worst case scenario. The user must go to an alternative mode, for example, the multi-tap for text writing. In short, the T9 system for mobile phones is very efficient. Nonetheless, both (if the word is not in the dictionary or it is not shown in first position) make the KSPC to be slightly grater than 1.

In order to implement this method in ambiguous VK with n keys (Tnk method), a NEXT key is required on the keyboard. Using the NEXT key, the user can see the next offered word. However, this can produce fatigue to the user and increase the number of user inputs. It would be preferable to use an automatic variant that generates a linear scan among the items of the list. This variant requires only an additional user input in order to choose the word of the list. The time scan between list items will be considered equal to $T_{scan.}$

In Tnk mode, the SPACE character has an additional functional: accepting the initially offered word. For this, SPACE character needs to not be integrated with another character in the same key, because the system would not be able to distinguish if the user is selecting the suggested word or if he(or she) is introducing a new character. Figure 8 shows the generic structure of a VK where the NEXT and the SPACE keys are separated. Through the visor the user recognizes the first suggested word by the system. If the SPACE key is pressed, the suggested word is accepted and the SPACE between words is automatically introduced. If the NEXT key is pressed, the system shows other suggested words in a new window where each word is highlighted through linear scan. In such a situation a user input makes the system introduce the outstanding word in the text. Nevertheless, if the scan of suggested words reaches the end of the list, it continues over VK in multi-tap mode, where, now, it is no longer necessary to select the SPACE key (the system does it automatically), because the word length is known and there are no typewriter errors.

The text entry rate and the number of user inputs for each character in Tnk mode is related to $\overline{t_K}$ y $\overline{UI_K}$. They represent the time necessary to select a key and the number of user actions to access it. In an ideal situation, where all sought words were always suggested by the VK in the first position, the average text entry time per character would be given by equation 4, where $t_{sp}$ is the access time to the Key with the SPACE function and we are assuming that the average length of a word in Spanish is 5 letters.

$$\overline{t_c} \to 5\overline{t_k} + t_{sp} \quad (4)$$

A similar result can be obtained for user input (equation 5). Here $KSPC_{sp}$ is keystrokes for character that space key requires.

$$\overline{UI_c} \to (5\overline{UI_k} + KSPC_{sp})/6 \quad (5)$$

An estimation of the values of $t_k$ and $UI_k$ can be obtained by the probability of one key, $T_{ij}$ (placed on row i, column j of VK). This probability is obtained by adding the probabilities of the letters, l, that the key contains, and the time employed by scanning to reach it, or $t(T_{ij})$ (equation 6).

$$\begin{aligned} \overline{t_k} &= \sum p(T_{ij}).t(T_{ij}) \\ \overline{UI_k} &= \sum p(T_{ij}).UI(T_{ij}) \\ p(T_{ij}) &= \sum_{\forall l \in T_{ij}} p(l) \end{aligned} \quad (6)$$

As the ambiguity of the keyboard increases, $t_K$ and the probability of finding the sought after word at first position decreases. This causes an increment in the key NEXT use. Apart from RED4L keyboard, and assuming that it's not possible to achieve a 100% hit rate on the dictionary, all keyboards have greater UIc than unambiguous ones. Also, it's suitable VKs where its $t_K$ parameter be much lower than $t_c$ for CONV. Thus, the time penalties, caused by dictionary misses, could maintain tc for ambiguous VKs under the one for CONV. Table 2 shows text entry time and number of user inputs per character for differents Vks operated by Tnk mode with a dictionary hit rate of 85%.

TABLE II. COMPARISON AMONG DIFFERENT AMBIGUOUS KEYBOARDS WITH A TSCAN = 1 IN TNK MODE.

| layout | tc | UIc |
|---|---|---|
| red4 | 3.15 | 1.4 |
| red6 | 4.30 | 2.79 |
| red9 | 4.39 | 2.79 |
| red12 | 4.94 | 2.77 |
| red16 | 5.02 | 2.70 |
| conv | 4.18 | 2 |

## V. AMBIGUOUS KEYBOARD WITH CHARACTER PREDICTION

Discarding multi-tap mode, because mathematical model results have shown that it has a worse performance

with respect to unambiguous keyboard, we consider a VK that uses a prediction list and a Tnk operation mode.

When scanning starts, the user goes on selecting keys, one after the other one. Each key selection makes the VK show the most likely characters associated to previous prefix in each item of prediction list. Notice that each key contains various characters and, therefore, a sequence of stroked keys generates a set of prefixes. Prediction list is made from a prefix table using this set. Scanning starts on the prediction list, when prediction is enabled, and continues on keys array if any item of prediction list is selected (this process is shown in figure 7).

When the user finishes selecting the keys that contain the character of the word that he or she wants to write, the window of VK shows the most likely word. Selecting a SPACE character accepts this word and introduces a space in the text. But if the word shown in the window is not the wished one, the user must select NEXT in order to VK shows the rest of the suggested words. In such a case, a new window that contains the words that match with key selected sequence is opened. Scanning continues on this new window item by item. If searched word is not in it, scanning continues on the keys array but now in multi-tap mode. Notice that the SPACE character could be in prediction list and, therefore, its selection causes the same effect that the user would choose the SPACE key in the keys array: acceptance of the suggested word that appears on window and introduction of a gap in the text.

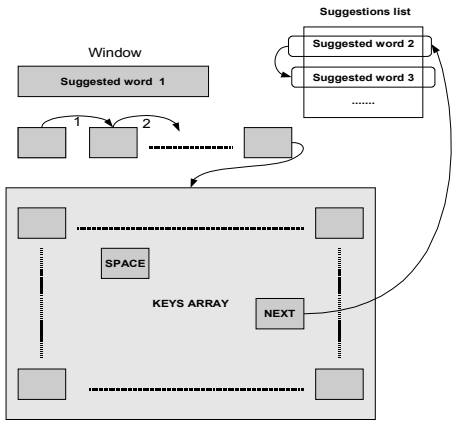


Figure 7. Ambiguous VK with character prediction and Tnk operation mode.

## VI. Simulation Results

In order to check the model's effectiveness, a program has been made in C that interact with the database MySQL, that houses prefix table. This database has been obtained from a dictionary with the 10,000 most likely Spanish words [11]. The program emulates VK operation and uses a sample text selected from a collection of documents of various types: sports, religion, etc., with a total of 960,180 words and an average of 4.91 characters per word. The mean hit rate in the dictionary is 0.85%.

For all experiments, we have used two parameters: prediction list length, ls, and prediction turn-on delay, d, that represents the number of needed inputs before prediction runs. We will also use similar reference of comparison results obtained from unambiguous keyboards with character prediction. Our interest is to demonstrate if there exists advantages using ambiguous keyboard instead unambiguous one, when they have character prediction. We will also focus on 4-keys VK (RED4), because it's the only the VK that, in Tnk mode, that improves unambiguous one.

### *A. Unambiguous virtual keyboard results*

Figures 8 and 9 show average text entry time and the average number of user inputs per character for unambiguous keyboard. .

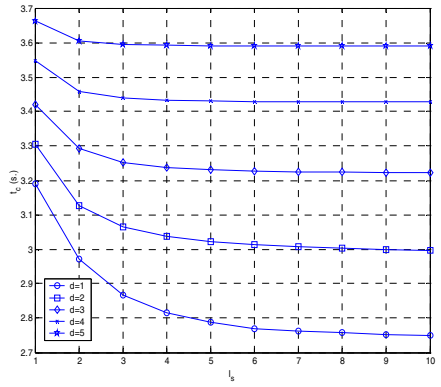


Figure 8. Average text entry time per character for unambiguous virtual keyboard with linear scanning of its prediction list.

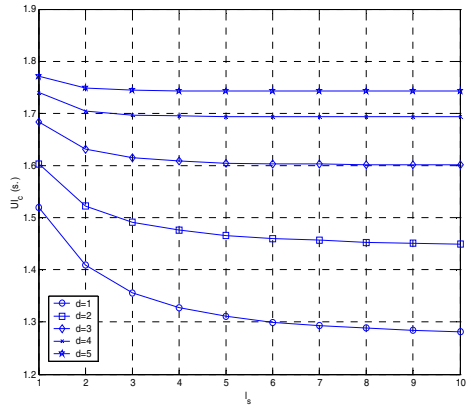


Figure 9. Average number of user inputs per character for unambiguous virtual key-board with linear scanning of its prediction list.

For unambiguous virtual keyboard the best choice is a prediction turn-on delay equal to 1 and a prediction list as

long as possible. With this selection, an unambiguous keyboard with character prediction obtains better performance that 4-key VK with a Tnk operation mode.

### B. *Ambiguous keyboard results*

Figure 10 shows average text entry time per character for different lengths of list and prediction turn-on delays with linear scanning of prediction list. The time $t_c$ decreases as delay increases for a $l_s = 1$. This situation is maintained until $l_s$ reaches a value equal to 4. From that value, if $l_s$ increases, the time $t_c$ behaves in an opposite way. This said, $t_c$ increases as delay does for high $l_s$ values.

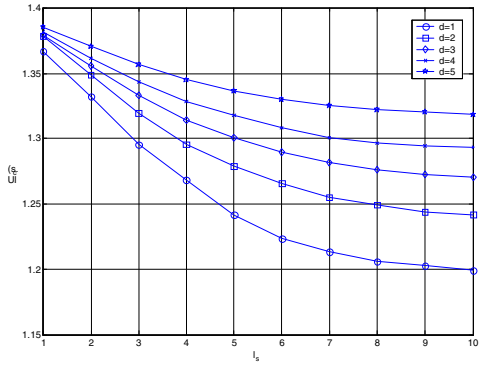


Figure 10. Average text entry time per character for 4-keys virtual keyboard (RED4) with linear scanning of its prediction list.

Figure 11 shows the average number of user inputs per character for 4-keys VK . For all possible values of d, $t_c$ decreases as ls increases. The number of user inputs increases as delay does. Notice that $UI_c$ improves with respect to ambiguous VK with Tnk (see table 2) operation mode. This is caused, on one hand, due to the fact that the user finds the character on the prediction list, although its probability be low. On the other hand, selecting a character on the list helps VK to do disambiguation, thus the probability to find the searched word on first place is high. In comparison to unambiguous one, this one has better performance.

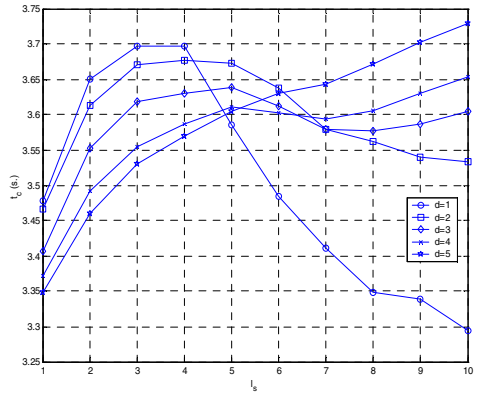


Figure 11. Average number of user inputs per character for RED4 with row-column scanning of its prediction list.

## VII. CONCLUSIONS

In this paper an exhaustive study of different methodologies to entry text using character prediction is shown. This study includes text entry speed and the number of inputs that a user has to generate. We've focused on VK operated by a external button.

The results demonstrate that there is no VK and method that improve both $t_c$ and $UI_c$. An unambiguous VK with character prediction, delay equal to 1, with a long prediction list and linear scanning is better to minimize $t_c$. On the other hand, a 4-keys VK with similar conditions for delay and list length minimizes $UI_c$.


## ACKNOWLEDGMENT

This project has been carried out within the framework of a research program: (p08-TIC-3631) – Multimodal Wireless interface funded by the Regional Government of Andalusia.


## REFERENCES


[1] John Arnott. Probabilistic character disambiguation for reduced keyboards using small text samples, volume Augmentative & Alternative Communication. Taylor & Francis, September 1992.

[2] Gregory W. Lesher, Bryan J. Moulton, and D. Jeffery Higginbotham. Techniques for augmenting scanning communication. ISAAC, 1998. A.B. Smith, C.D. Jones, and E.F. Roberts, "Article Title", *Journal*, Publisher, Location, Date, pp. 1-10.

[3] Karin Harsbush and Michael Khn. An evaluation study of two-button scanning with ambiguous keyboards. AAATE, 2003.

[4] Edmund F. Lopresti Richard Simpson, Heidi H. Koester. Selecting an appropriate scan rate: the .65rule. RESNA proceedings, 2006.

[5] Richard Simpson and Heidi Horstmann. Adaptive one-switch row-column scanning. IEEE Transactions on rehabilitacion engineering vol7. N 4, 1999.

[6] Chris Bloor Paul Gnanayutham and Gilbert Cockton. Soft keyboar for the disabled. ICCHP, Springer-Verlag, 2004.

[7] T. Bellman and I. MacKenzie. A probabilistic character layout strategy for mobile text entry. In Proc. Graphics Interface, pages 168-176, 1998.

[8] Bryan Moulton Gregory Lesher, Jeery Higginbotham. Techniques for automatically updating scanning delays. In Proceedings of the RESNA 2000 Annual Conference, pages 8587, 2000.

[9] I. Scott MacKenzie Miika Silfverberg and Panu Korhonen. Predicting text entry speed on mobile phones. CHI, 2000.

[10] I. Scott MacKenzie, Hedy Kober, Derek Smith, Terry Jones, and Eugene Skepner. Letterwise: prex-based disambiguation for mobile text input. UIST Orlando 01, 2001.

[11] Ramn Almela, Pascual Cantos, Aquilino Sanchez, Ramon Sarmiento, and Moises Almela. Diccionario y estudios lexicos y morfologicos. Universitas SA,