The Metropolis Keyboard – An Exploration of Quantitative Techniques for Virtual Keyboard Design

Shumin Zhai **Michael Hunter** **Barton A Smith**

IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120
+1 (408) 927 1112
{zhai, basmith}@almaden.ibm.com, michael@hunter.org

ABSTRACT

Text entry user interfaces have been a bottleneck of non-traditional computing devices. One of the promising methods is the virtual keyboard on touch screens. Various layouts have been manually designed to replace the dominant QWERTY layout. This paper presents two computerized quantitative design techniques to search for the optimal virtual keyboard. The first technique simulated the dynamics of a keyboard with "digraph springs" between keys, which produced a "Hooke's" keyboard with 41.6 wpm performance. The second technique used a Metropolis random walk algorithm guided by a "Fitts energy" objective function, which produced a "Metropolis" keyboard with 43.1 wpm performance.

The paper also models and evaluates the performance of four existing keyboard layouts. We corrected erroneous estimates in the literature and predicted the performance of QWERTY, CHUBON, FITALY, OPTI to be in the neighborhood of 30, 33, 36 and 38 wpm respectively. Our best design was 40% faster than QWERTY and 10% faster than OPTI, illustrating the advantage of quantitative user interface design techniques based on models of human performance over traditional trial and error designs guided by heuristics.

Keywords

Graphical keyboard, soft keyboard, virtual keyboard, on screen keyboard, text entry, text input, mobile computing, mobile devices, pen based computing, ubiquitous computing, pervasive computing, Metropolis method.

INTRODUCTION

Pervasive computing on small, mobile or wall size devices are one of the most important trends in the evolution of computing technology. These devices include personal digital assistants (PDAs), tablet computers, electronic whiteboard or communication devices such as two-way pagers and mobile phones. As an extension of the Internet, these devices may free us from desktop or laptop computers, moving us to a future of pervasive information.

One obstacle to the progress of pervasive computing is text entry. For example, a recent study [13] clearly demonstrated the value of tablet computers in home environments, but the lack of efficient text input techniques in these tablet computers made many common applications, such as chat, email, or even entering a URL, very difficult.

Currently there are many types of text entry methods for mobile devices, including reduced physical keyboards, handwriting recognition, voice recognition, and virtual keyboards. Each has critical usability shortcomings.

Physical keyboards. There are two ways to reduce the size of physical keyboards. One is to shrink the size of each key. This is commonly seen in electronic dictionaries. Typing on these keyboards is slow and difficult due to their reduced size. The other method is to share each physical button with multiple letters, as in a telephone pad. Resolving ambiguity is a difficult challenge to the usability of such a method [15].

Handwriting. Although reducing error rate has been the major goal in handwriting recognition, the ultimate bottleneck is the human handwriting speed limit. It is very difficult to write legibly at a fast speed. Our informal tests showed that a well practiced user of Graffiti on a PalmPilot PDA can write about 20 words per minute (wpm), far slower than a well practiced typist on a physical keyboard. 20 wpm is good enough to enter names and telephone numbers, but not satisfactory for online chatting or email.

Voice recognition. Speech has been expected to be a compelling alternative to typing for text input. Despite the progress made in speech recognition technology, however, a recent study [7] showed that the effective speed of text entry by continuous speech recognition was still far lower than that of the keyboard (13.6 vs. 32.5 corrected wpm for transcription and 7.8 vs. 19.0 corrected wpm for

in *Proceedings of ACM Symposium on User Interface Software and Technology (UIST 2000),* November 5-8, 2000, San Diego, California. pp 119-128.

Copyright (c) ACM

composition). Interestingly, the study also revealed many more human-factors issues that were not well realized before. For example, many users found it " harder to talk and think than type and think" and considered the keyboard to be more " natural" than speech for text entry.

The present paper focuses on virtual keyboards displayed on screen and tapped with a stylus or a finger. The most commonly used virtual keyboards today adopt the traditional QWERTY layout (Figure 1). As we will see the QWERTY layout is particularly inefficient for stylus tapping. There is a long, fascinating, and controversial history of inventions and studies of the layout of the physical keyboard, well documented by Yamada [18]. Various attempts of replacing QWERTY with more efficient layouts, such as Dvorak, have been made but the performance gain has not justified the cost of retraining the great number of users [3, 14, 18]. The opportunity of designing an optimized layout for the physical keyboard seems forever lost.

This may not be true for the virtual keyboard. First, it is not yet too late to form a new layout standard for the virtual keyboard. Second, the ten-finger touch typing skills on a physical keyboard do not necessarily transfer to on screen stylus tapping on the same layout [19]. The perceptual, memory, and motor behavior of using a virtual keyboard is sufficiently different from that of a physical keyboard to justify a different design. Third, the efficiency of a virtual keyboard depends on very different, if not opposite, mechanisms from that of a physical keyboard. For a physical keyboard, one key factor influencing its efficiency is the frequent alternation between the two hands. This means that frequent adjacent letter pairs (digraphs) should be on the opposite sides of the keyboard. For a virtual keyboard tapped with a stylus, however, the frequent digraphs should be close to each other so the hand does not have to travel much.

PERFORMANCE MODELLING OF VIRTUAL KEYBOARD

To minimize the hand travel on a virtual keyboard, the transitional frequencies from one letter to another in a given language have to be taken into account. The goal is to lay out the letters so that the statistical total travel distance is the shortest when typing in that language. This means that the most frequent keys should be located in the center of the keyboard and frequently connected letters (such as T and H) should be closer to each other than the less frequently connected letters.

Figure 1: The virtual keyboard in a PDA, with a QWERTY layout (30 wpm)

MacKenzie and colleagues [16], [11] were the first to use a quantitative approach to model virtual keyboard performance. Their model predicts user's performance by summing the Fitts' law movement times between all digraphs, weighed by the frequencies of occurrence of the digraphs. The use of Fitts' law made it possible to estimate performance in absolute terms, giving us a comparison to speed we are familiar with, such as 60 wpm for a good touch typist.

According to Fitts' law (Figure 2), the time to move the tapping stylus from one key i [1] to another j for a given distance (D_{ij}) and key size (W_j) is

$$MT = a + b\,Log_2(D_{ij}/W_j + 1), \qquad (1)$$

where a and b are empirically determined coefficients. In order to be able to make comparisons to the results in [11], we choose the same values a = 0, b = 1 / 4.9. In other words, we consider the Fitts' index of performance (IP) [4] to be 4.9 bits per second. We will return to this choice of this parameter later.

If the frequency of letter j to follow letter i (digraph ij) among all digraphs is P_{ij}, then the mean time in seconds for typing a character is:

$$\bar{t} = \sum_{i=1}^{27}\sum_{j=1}^{27} \frac{P_{ij}}{IP}\left[Log_2\left(\frac{D_{ij}}{W_i}+1\right)\right] \qquad (2)$$

Assuming 5 characters per word (including space key), this equation allows us to calculate tapping speed in words per minute (wpm = 60 / 5 $\bar{t}$).

Note that a special case has to be made for equation (2) when $i = j$. This means tapping on the same key successively (e.g. oo as in " look"). In this case, the second term in equation (1) is 0 but a is set at 0.127 sec. Previous authors [19] [11] have used both 0.127 or 0.135 second. We choose 0.127 because it was closer to what we

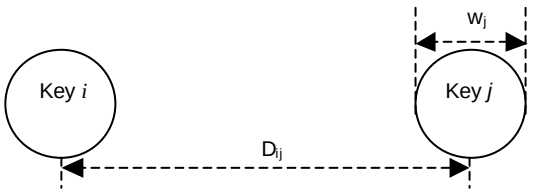

Figure 2. For given key size and distance, Fitts' law predicts the time of tapping from one key to another

[1] i and j here represent any pair of keys from A to Z and the space key.

measured (7.8 repeats on the same location). The influence of this number is small, however, due to the low frequency of such cases.

Equation (2) only estimates the movement time of tapping on a virtual keyboard by an "expert". A novice or intermediate user has to visually search for the destination key before tapping on it. In that sense, Equation (2) only predicts the upper bound of a user's performance [16]. However, the Fitts' law coefficient in the model is based on average human tapping performance. Some users therefore could surpass this "upper bound".

Digraph frequency

The digraph frequency P_{ij} in equation (2) is numerically calculated by the ratio between the number of i-j digraphs and the total number of digraphs in an English text corpus. One commonly used digraph table is by Mayzner and Tresselt, extracted from an English text corpus of 87,296 characters [12]. There are two shortcomings to this table. First it is not clear if their corpus still reflects the usage of English in today's digital media. Second, they limited their selection of words to those of length 3 to 7 letters, which eliminated some of the most common words such as I, in, on, is, at, to, of, if.

We hence constructed two digraph tables ourselves. One was from a text corpus consisting of news articles from sources such as the San Jose Mercury News, New York Times, and the LA Times. The articles covered a range of topics from technical to social. The size of the corpus was 101,468 characters (no spaces). The second 1364497-character corpus was gathered from logs of six on line chat rooms. The names of the chat rooms were Teen, Atheism, ChristianDebate, Myecamp, CityoftheGreats, and MaisonlkkoguRPG. The text consists of very informal conversations. Most input strings are less than 80 characters, and many are not complete sentences. They are frequently a one or two word answer to a question or comment about a previous statement. Capitalization at the beginning and periods at the end of sentences are frequently missing. We included only those records that appeared to be typed by a person. Computer-generated headers, etc. were deleted.

Due to the space constraint of this paper, we have to publish our digraph analyses elsewhere. For the purpose of the virtual keyboard design, however, the differences among the three corpuses were small, despite the very different styles of English used. This means that a keyboard layout does not have to be tuned to a specific area of application.

To be consistent with the literature, we continued to use the Mayzner and Tresselt table in evaluating existing keyboard designs. We used all three tables in designing the Metropolis keyboard presented later in this paper.

EXISTING LAYOUTS AND THEIR PERFORMANCE ESTIMATION

This section reviews various existing layouts of virtual keyboards and applies the Fitts-digraph model (2) to estimate the performance of these layouts. This is necessary for two reasons. First, other than informal arguments and promotional data, some of the existing layouts have never been scientifically evaluated. Second, previously published estimates of QWERTY and OPTI keyboards in the literature [11] [19] are incorrect.

The Performance of QWERTY layout

The space key in the QWERTY layout has a much greater length than the rest of the keys. The Fitts' law distance between a character key and the space key varies depending on what point of the space key is tapped. MacKenzie and Zhang [11] [19] [9] used a "sub-optimal" model to handle the space key, which divided the space key into multiple segments, each was equal in length to a regular character key. For each character-space-character "trigraph", they chose the segment of the space key that yielded the shortest total distance of character-space-character path. They then calculated the two Fitts' law times of character-space and of space-character according to the distances along that path. Finally, they summed the Fitts' tapping times of all of character-space-character trigraphs, weighted by the probability of each trigraph occurrence. Using this approach, they estimated the QWERTY performance of 43.2 wpm.

Following the same methodology, we could not replicate their result. Upon close examination, we found a subtle error in MacKenzie's and Zhang's calculation of the probability of each character-space-character trigraph. Taking the combination of i-space-j as an example, they incorrectly used the $P_{i\text{-}space} \times P_{space\text{-}j}$ to calculate the probability of this path ($P_{i\text{-}space\text{-}j}$). Note that $P_{space\text{-}j}$ is the probability (or frequency) of the transition at any given tapping to be from the space key to the J key. It is not the *conditional* probability of $P_{space\text{-}j/space}$ ($= P_{space\text{-}j} / P_{space}$.) needed to calculate the probability of two serial events. The correct calculation should be:

$$P_{i\text{-}space\text{-}j} = P_{i\text{-}space} \times P_{space\text{-}j/space} = P_{i\text{-}space} \times P_{space\text{-}j} / P_{space,} \quad (3)$$

where $P_{i\text{-}space}$ is the probability of i-space digraph at any given tapping, $P_{space\text{-}j}$ is the probability of space-j at any given tapping, $P_{space\text{-}j/space}$ is the conditional probability of space-j, given the last tapped key is space, and P_{space} is the probability (frequency) of the space key.

Using this corrected $P_{space\text{-}j/space}$ calculation but following the same "sub-optimal" methodology as in [19], we found the performance of QWERTY layout to be 30.5 wpm.

In order to ensure the correctness of our estimation, we also applied two much simpler methods, one conservative and

one liberal, both involved only digraphs and avoided character-space-character trigraphs. By the conservative method, character to space distance was always measured to the center of the space key. Obviously the result of this should be lower than the estimation of sub-optimal model. Indeed, the performance was 27.6 wpm calculated by this method. By the liberal method, the distance between the space key and any character key was always measured along the shortest (vertical) line from the character to the space key. Due to the " free warping" effect – the stylus goes into the space key from one point and comes out from another point of the space key without taking any time. This should produce a higher estimate than the sub-optimal model. Indeed, we found the tapping speed to be 31.77 wpm by this method.

In conclusion, the performance of a QWERTY keyboard is about 30 wpm, assuming the user always taps on the portion of the space key that minimizes the character-space-character total path. If the user does not plan one key ahead, the performance would be lower than 30 wpm.

The Performance of the OPTI layout

MacKenzie and Zhang [19] [11] designed a new layout, dubbed OPTI (Figure 3). They first placed the 10 most frequent letters in the center of the keyboard. Then assiging the 10 most frequent dgraphs to the top 10 keys, they placed the remaining letters. The placement was all done by trial and error. They later made a further improved 5X6 layout, shown in Figure 4. For convenience, we call the 5X6 layout OPTI II in this paper.

There are four space keys in both OPTI keyboards, evenly distributed in the layout. The user is free to choose which one to tap. The optimal choice depends on both the

Figure 3. MacKenzie's and Zhang's OPTI layout

Figure 4 The improved OPTI layout in a five by 6 layout (OPTI II) [18] (38 wpm)

preceding and following key to the space key. For example, for the sequence of M - space – V (Figure 4), the upper right space key is the best choice. However, the upper right space key is not the optimal choice if the tapping sequence is M - space -Y. In practice, the use of the optimal space key ranged from 38% to 47%, depending on the user's experience [11].

Assuming optimal choice of space keys, MacKenzie and Zhang predicted 58.2 wpm performance of the OPTI keyboard and 59.4 wpm of the OPTI II layout. These were surprisingly high performance scores that we could not replicate. As in their QWERTY performance estimation, MacKenzie and Zhang [19] [11] used the character-space-character " trigraph" approach to handle the multiple space keys. They made the same probability miscalculation of the trigraphs on the OPTIs as they did on the QWERTY.

We re-calculated the performance of the OPTI II keyboard. The first approach was the same as that of MacKenzie and Zhang, except we used the corrected conditional probability in calculating trigraph probabilities. Our result of the OPTI II performance is 40.3 wpm. This result was based on the assumption that the user always used the optimal choice of space keys.

Our second approach took the non-optimal choice of space keys into account. We assumed that the user would make use of the optimal space key 50% of time, which was still higher than the highest actual rate. For the rest of the time, the average distance from the character key to the three non-optimal space keys was used. By this approach, we found the OPTI II performance to be 36 wpm.

In conclusion, the OPTI II performance should be between 36 and 40.3 wpm, depending on the optimality of the space key choice. If we take 38 wpm as a fair (but optimistic) estimate, this is a 27% improvement over QWERTY performance (about 30 wpm). 38 wpm is beyond what legible handwriting could achieve.

MacKenzie and Zhang conducted an experiment to investigate how quickly users could reach the predicted performance. In their test, participants reached 44.3 wpm after 20 sessions of text entry, each for 45 minutes, on the OPTI design [11]. This is higher than our predicted performance of 38 wpm. We think this disparity is due to the low value of Fitts' law index of performance (*IP*) used in equation (2). Originated in [10], 4.9 bits/s might be an overly conservative estimate of human tapping performance. For two adjacent keys (1 bit), 4.9 bits/s means 204 ms per tap. This is a much longer than what we measured (average 160 ms). The rate of index of performance reported in the literature for tapping varied widely. For example, Fitts' original report was 10.6 bits/s [4]. This rate dropped to 8.2 bits/s after adjusting for the effective width and for the Shannon-MacKenize formulation [8], but it is still much higher than 4.9 bits/s. In order to make comparisons to previous studies, we

maintain the 4.9 bits/s index of performance used in (2). One should be aware, however, that all predicted performance scores in this paper can be proportionally scaled according to the IP rate. For example, if we use 6 bits/s instead of 4.9 bits/s. The OPTI II performance would be 38 x 6/4.9 = 46.5 wpm.

The Performance of the FITALY keyboard

The FITALY keyboard (Figure 5) is a commercial product by Textware Solutions. The design rationale behind this layout included center placement of more frequent keys, dual double sized space keys, and the consideration of digraph frequencies [17].

In a loosely controlled contest (self reporting with a witness, best performer rewarded with a prize), Textware Solutions collected 34 entries of text entry speed on a Palm Pilot PDA, with 19 contestants using FITALY, 9 using Graffiti, and 6 using QWERTY. FITALY received the highest average score (44.4 wpm), followed by QWERTY keyboard and Graffiti handwriting (both 28.2 wpm) [17]. Note that these scores were collected from motivated contestants.

Applying the Fitts-digraph performance model (2), we did a formal estimation of the FITALY layout. In our calculation, the two double sized space keys were treated differently from other regular keys. First, the width of the space keys was considered twice the size of a regular key in the Fitts' law calculation. This clearly was an overestimate when the movement was primarily vertical. Second, there was again the issue of which space key to use in calculating distances. Two methods were used to deal with this issue. The first always used the closest space key to each character, with " free warping" between the two space keys. By this method, the FITALY keyboard performance was estimated as 37.07 wpm. The second method used the shortest character-space distance 75% of the time. The rest of the time the further space key was used in calculating distance. By this method, performance of 35.2 wpm was found.

In summary the performance of the FITALY keyboard is about 36 wpm, far more efficient than QWERTY, as the company advertised, but less efficient than OPTI II is.

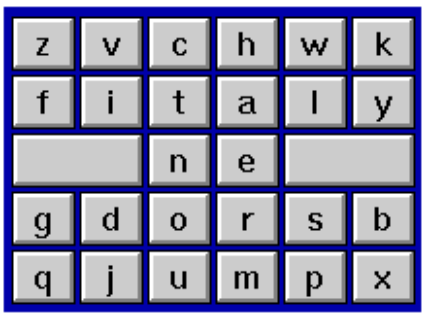

Figure 5. The FITALY keyboard (36 wpm) [17] Copyright Textware Solutions 1996-1998

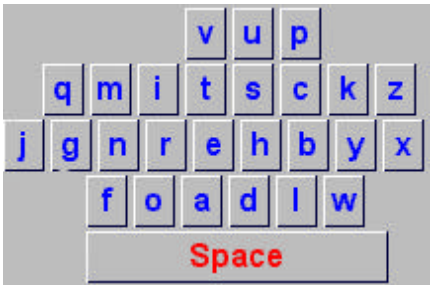

Figure 6. The Chubon Keyboard (33 wpm)

The Performance of the Chubon keyboard

Figure 6 shows the Chubon keyboard layout [5]. Using the same approach as in the case of the QWERTY layout, we estimated its performance to be 33.3 wpm, assuming free warping. If we assume the user always taps at the center of the space key, the performance will be 32 wpm. Both estimates are slower than OPTI and FITALY but still faster than QWERTY.

IN SEARCH OF THE OPTIMAL - COMPUTERIZED QUNTITATIVE DESIGN

We have reviewed three designs of virtual keyboards alternatives to QWERTY. The best so far is MacKenzie's and Zhang's OPTI II keyboard. Has the OPTI II reached the optimal? Can we design an even better keyboard? The existing designs are all based on manual trial and error approaches, with the help of letter and digraph frequency tables. Given the great number of possibilities, human manual exploration can only try out a small fraction of arrangements.

In a departure from the manual exploration approach, we decided to explore computerized techniques to design the optimal virtual keyboard. The first method that comes to mind could be an exhaustive algorithmic searching approach. However, the complexity of that search - O(n!) - is approximately 10^{28}, an impossible number for computing. Thus, we designed and implemented two systematic, physics-based techniques to search for the optimal virtual keyboard. This section presents the methodologies and results of these two techniques.

The Dynamic Simulation Method

As analyzed earlier, the goal of a good keyboard design is to minimize the statistical travel distance between characters. The more frequent digraphs should be closer together than less frequent digraphs. In order to achieve this goal, we first designed a dynamic system technique. Imagine a spring connecting every pair of the 27 keys whose initial positions were randomly placed with spaces between the keys. The elasticity of the springs, when turned on, were proportional to the transitional probability between the two keys so that keys with higher transitional probability would be pulled together with greater force. In

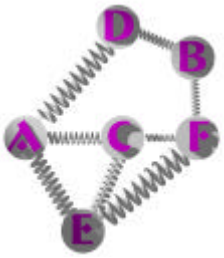

Figure 7. (Part of) the dynamic simulation model: frequent digraphs are connected with stronger springs

addition, there is viscous friction between the circle shaped keys and between the key surface and the table surfaces. The steady state when all keys are pulled together forms a candidate virtual keyboard design. Figure 7 illustrates one part of this dynamic system model.

Fortunately, we did not need to build physical models to create the spring-viscosity-mass dynamic systems. Instead we used a mechanical simulation package (Working Model) to simulate it. In the simulation, the springs were "virtual" . They did not stop other objects passing through them, hence preventing the springs from being tangled.

The final positions of the keys might still not be at the minimum tension state, because some keys could block others from entering a lower energy state. Two methods were used to reduce the deadlock or local minimum states. First, we experimented with different initial states, which had very significant impact to the end result. Second, each spring had an extended segment - a strut - that held the keys apart so other keys could be pulled through these gaps to reach a lower level of tension. The length of this segment was manually adjusted in the dynamic simulation process. At the end of each simulation cycle, we reduced the length of the adjustable struts to zero so all the keys were pulled against each other, forming a layout of a virtual keyboard. The performance of the design was then calculated according to equation (2) and compared with known results. When not satisfactory, the layout could be 'stretched' out to serve as another initial state for the next iteration of the same process. The iteration was repeated until a satisfactory layout is formed. Figure 8 shows the best layout we achieved with this approach. To capture the

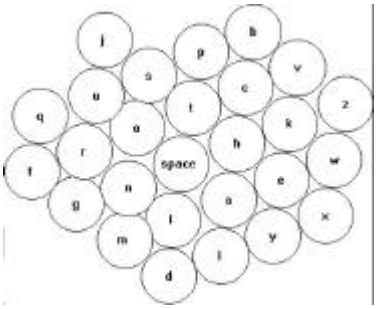

Figure 8. Hooke's Keyboard (41.6 wpm)

gist of the spring simulation technique, we call the best design achieved through this method Hooke's keyboard (after Hooke's Law). The performance estimate of the Hooke's keyboard shown in Figure 8 is 41.6 wpm, higher than the best previous design (OPTI II, 38 wpm).

Fitts' Energy and the Metropolis Method

The idea of minimizing energy, or tension, in the keyboard layout brought us to explore a better known optimization method - the Metropolis algorithm. The Metropolis algorithm is a Monte Carlo method widely used in searching for the minimum energy state in statistical physics [1]. If we define equation (2) as " Fitts' energy" , the problem of designing a high performance keyboard is equivalent to searching for the structure of a mo lecule at a stable low energy state. Applying this approach, we designed and implemented a software system that did a "random walk" in the virtual keyboard design space. In each step of the walk, the algorithm picked a key and moved it in a random direction by a random amount to reach a new configuration. The level of Fitts' energy in the new configuration, based on equation (2), was then evaluated. Whether the new configuration was kept as the starting position for the next iteration depended on the following Metropolis function [2]

$$W(A \to B) = e^{-\Delta E / kT} \quad if\ \Delta E > 0$$
$$= 1 \quad if\ \Delta E \le 0 \qquad (3)$$

In equation (3), W (A-B) was the probabilty of changing from configuration A (old) to configuration B (new), ΔE was the energy change, k was a coefficient, T was "temperature" , which could be interactively adjusted. The use of equation (3) makes the Metropolis method superior to our previous spring model because the search does not always move towards a local minimum. It occasionally allowed moves with positive energy change in order to be able to climb out of a local energy minimum.

Again, the initial state where the random walk starts from had a significant impact on the search process. An existing good layout stretched over a larger space was used as an initial state.

In addition to the automatic random walk process itself, we also applied interactive "annealing" as commonly used in the Metropolis searching process. The annealing process involved bringing "temperature" – T in equation (3) through several up and down cycles. When temperature was brought up, the system had a higher probability of moving upwards in energy and jumping out of local minima. When temperature was brought down, the system decended down to a lower energy level. This annealing process was repeated until a sufficiently efficient keyboard layout was found. Figure 9 to 11 are snapshots from the Metropolis random walk process in one annealing cycle.

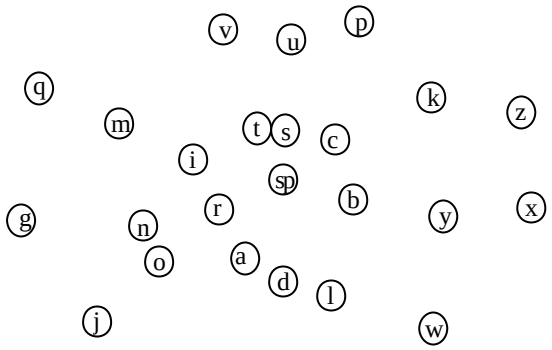

Figure 9. Early stage of Metropolis random walk. The system is at a high energy state, moving towards lower energy state.

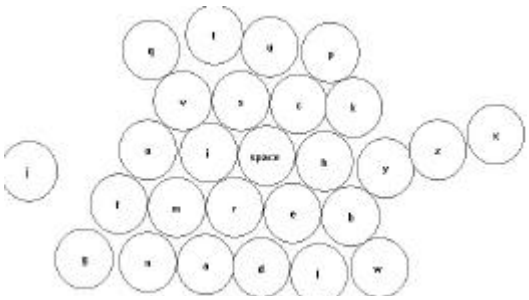

Figure 10. Middle stage of Metropolis random walk. Keys have been descended to a lower energy state. They are getting packed.

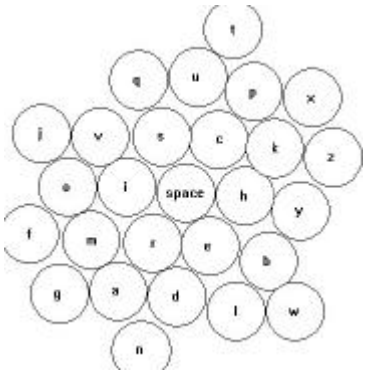

Figure 11. Later stage of Metropolis random walk. Keys moved to a lower still energy state.

We call layouts produced by this process Metropolis keyboards. Various layouts with similar performance were produced. Figure 12 shows one of them. Particularly interesting with this layout is that the vowels are connected symmetrically. These vowel keys may potentially serve as "landmarks", making the layout more structured and easier to remember. In Figure 12, we replaced the circle shapes used in the design process with hexagons. Each hexagon

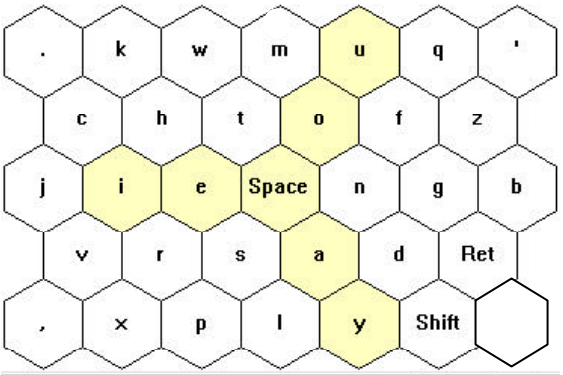

Figure 12. The Metropolis keyboard (43.1 wpm)

encapsulated the circle it replaced and filled the gaps between the circles, making more efficient use of the total space.

The performance of this layout is 43.1 wpm[2] based on the Mayzner and Tresselt digraph table. This is over 40% improvement over QWERTY and more than 10% improvement over OPTI II. When we applied our digraph tables constructed from current news and chat room text corpuses, the performance is 42.2 wpm and 42.3 wpm respectively.

Different from the manually designed previous layouts, neither the Hooke's nor the Metropolis layouts came in a rectangular shape that exactly fits on part of a PDA screen. However, this does not pose a problem. So far we have only considered the alphabetical keys. A fully functional keyboard needs punctuation and other auxiliary keys. These keys can be packed around the alphabetical keys to form a total rectangle shape, as we partially did in Figure 12. We could also apply the Metropolis method to search for the best design to fit any shape such as a rectangle. We could also possibly constrain the design to a triangle shape in order to fit the lower right corner of a mobile computing device. This may make it easier to tap for right handed users.

Multiple Space Keys and Varying Key Sizes

Both the OPTI keyboard and the FITALY keyboards used more than one space key to accommodate the high frequency of space in English writing (Figure 13). We decided against that for the following reasons. First, multiple space keys take more space from the real estate

[2] In a brief preliminary publication [6], we reported 43.7 wpm performance for a different Metropolis keyboard. This was due to a less conservative measurement used in calculating the width of each key. Instead of the diameter of the inscribed circle of a hexagon, we previously used the average of diameters of the inscribed and circumscribed circles of a hexagon.

available to regular keys, which may reduce the total efficiency of the keyboard. Second, as revealed by our analysis, the performance of a multiple space key keyboard highly depends on the optimal choice of the space key, which requires planning one key ahead of tapping. Third, the space key is not the only one with high frequency (Figure 13).

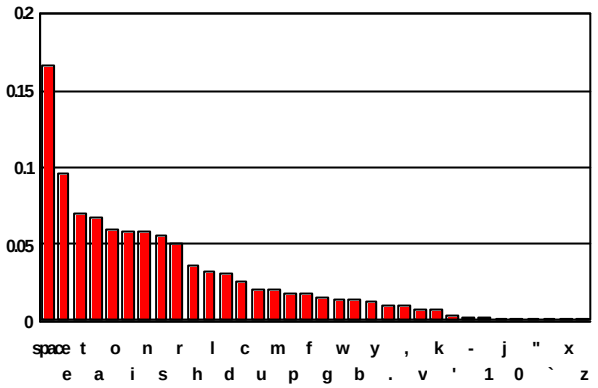

Figure 13. Relative character frequency distribution in our English new corpus

Related to the multiple space key issue is the size of the space key. Should the space key be given a greater size than other keys? Should other more frequent keys also be given greater size? According to Fitts' law tapping time is related to both distance and target size. We have optimized the statistical distance in order to reduce tapping time. By the same principle, shouldn't we also optimize the relative key sizes so more frequent keys are given a greater share of the real estate?

We have indeed explored the issue of varying key sizes but we have not come to a positive conclusion. There are at least four issues to resolve. First, the optimization of both size and distance is much more complex. One of the complicating factors is that unequal sized keys cannot be as tightly packed and still be optimized in positional layout. We have made several versions of search algorithms to optimize both the key sizes and the position layout, but so far we have not produced a keyboard that had higher performance than the equal size Metropolis keyboard shown in Figure 12.

Second, from Fitts' law point of view, frequently used keys should be given a greater size. However, these frequent keys also should be placed toward the center of the keyboard. Crossing these bigger sized keys to reach other keys introduces a performance penalty.

Third, there is an asymmetrical effect to size gain and loss in Fitts' law. To reciprocally tap on the two adjacent targets shown in Figure 14, the performance gain of tapping the enlarged right target is less than the loss of tapping the reduced left target. Figure 15 illustrates tapping time from

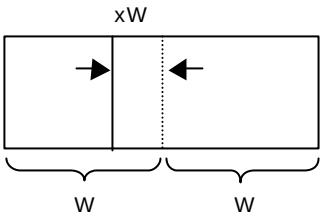

Figure 14. Asymmetrical Fitts' targets, with asymmetry coefficient factor x

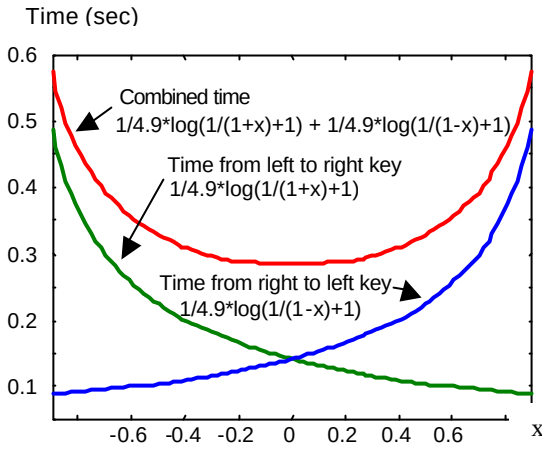

Figure 15. Tapping time from left to right key, vice versa, and the sum of the two as a function of the asymmetry coefficient x. The lowest sum occurs when x = 0, i.e. the two targets are equal in size.

the left to right and vice versa. As we can see, as the asymmetry factor x changes from 0 to positive, the time reduction from the left to the right target does not compensate for the time increase in the reverse direction. The lowest sum of the two is when x = 0, assuming equal frequency of entering the left and the right target.

Fourth, even if we found a more efficient layout with varying key sizes, there could be a cost of varying control precision depending on which letter is being typed. The loss of consistency in control precision may be detrimental.

In summary, varying size remains an open problem in user interface design, although the factors we have considered suggest against it.

IMPLEMENTATION

We have implemented the virtual keyboard both in Java and in Windows (Figure 16). Here we do not have space to discuss the implementation details. One interesting feature with the Metropolis keyboard is that the letters of some common parts of a word or an entire word, such as "the", are connected. Experienced users might be able to stroke through these letters instead of tapping on each of them. Such a strategy not only may save time but also enriches the input gestures.

Figure 16. A Metropolis keyboard

Has our exploration achieved the best possible design? We have run a large number of simulations but all reached approximately 43 wpm performance, suggesting we are very close, if not at, the best possible. However it is theoretically interesting to estimate the lowest upper bound performance. To that end, we have produced three reference points, based on three physically impossible layout designs. The first was that all keys were co-located in one spot, hence the user only needed to tap on the same spot. With such a hypothetical "keyboard", the estimated performance was 95 wpm. The second estimate assumed the next key needed was always next to the current key, requiring tapping with Fitts ID = 1 bit. Performance dropped to 59 wpm with this assumption. In the third estimate, when calculating Fitts' law performance for any key, all the rest of the keys were optimally placed according their digraph frequency to the current key. The performance of this more realistic but still impossible keyboard was 53 wpm, about 10 wpm faster than our best design. Finding the lowest upper bound of virtual keyboard performance is another open research issue.

DISCUSSION: USER INTERFACE DESIGN TECHNIQUES

Although sharing the same ultimate goal, user interface design and user interface evaluation are traditionally two entirely separate processes involving totally different methodologies. User interface evaluation tends to be measurement based while user interface design tends to be intuition, heuristics, and experience based. The design exploration presented here is a departure from that norm. First, the design process was quantitative and computerized. Second the design process was integrated to the highest degree with evaluation – every step of the design space search was guided by the evaluation function. Third, the quantitative design process was based on previous evaluation research, particularly the work on Fitts' law. Without Fitts' law, we could still construct an evaluation function simply based on the digraph frequency and travel distance, so we could know the relative superiority of one layout to another based on statistical distance. We would not, however, be able to relate the statistical distance to user performance. With Fitts' law modeled in the evaluation function, we could instantly estimate the eventual average user performance and compare it to known benchmarks (such as 40 wpm).

FUTURE WORK

A UI design is not complete without user studies. A user study merely to confirm the performance predictions presented here, however, would not be very informative. First, the very foundation of our design, Fitts' law, has already been repeatedly tested. Second, previous user studies have shown that users could indeed master a new layout and eventually reach the performance level of over 40 wpm on the OPTI design [11]. The more meaningful experiments would be on the learnablity of the keyboard. How quickly can users reach the level of Fitts' law performance? How do they learn the keyboard layout? Do they learn the paths of words or do they learn the positions of the keys? Or is it a combination of both? In the early learning stage, users have to scan the keyboard to find a particular key. Do they scan randomly or do they scan systematically? The Metropolis keyboard is designed such that the next letter key in a word is statistically close to the current letter key. Does it help to instruct users to scan by degree of vicinity? There are many fascinating cognitive issues here to be investigated. In short, our future work will switch from the "physics" to the "psychology" of the virtual keyboard.

CONCLUSIONS

Motivated by the increasing importance of pervasive computing devices and built upon the work of MacKenzie and colleagues [16] [11] [19], this paper explored the design of an optimal virtual keyboard and made the following contributions. First, the paper thoroughly and quantitatively analyzed the expert performance of existing virtual keyboards and corrected erroneous performance estimations of virtual keyboards in the literature. We found the performance of QWERTY, CHUBON, FITALY and OPTI keyboard to be in the neighborhood of 30, 33, 36 and 38 wpm respectively. Second, we introduced two computerized, quantitative techniques to virtual keyboard design. One technique used physical simulation of digraph springs and produced a "Hooke's" keyboard with 41.6 wpm performance. The other method used the Metropolis random walk algorithm, guided by the "Fitts' energy" object function. This method produced a Metropolis keyboard with 43.1 wpm performance, which was more than 40% faster than QWERTY keyboard. The 43.1 wpm was based on a very conservative assumption of 4.9 bits/s Fitts' law IP and can be scaled up with the IP value[3]. Third, the paper illustrates the benefits of quantitative design combined with the knowledge of human performance

[3] If we assume IP = 6 bits/s, then the performance of the Metropolis keyboard is 52 wpm.

modeling over traditional UI design methods based on manual trial and error and heuristics. We demonstrated the importance of quantitative techniques and basic human performance modeling to the field of user interface research.

ACKNOWLEDGMENTS

We thank Teenie Matlock, Paul Maglio, Eser Kandogan, Alison Sue, Rich Gossweiler, and other colleagues at the IBM Almaden Research Center for their input and assistance. Allison E. Smith helped us collecting the Internet relay chat corpus. We particularly would like to thank Dr. I. Scott MacKenzie for sharing his spreadsheet model and for numerous fruitful discussions.

REFERENCES

1. Binder, K. and D.W. Heermann, *Monte Carlo Simulation in Statistical Physics*. 1988: Springer-Verlag.
2. Coddington, P., *Monte Carlo Simulation for Statistical Physics*, 1999: http://www.npac.syr.edu/users/gcf/cps713montecarlo/node24.htm.
3. Cooper, W.E., ed. *Cognitive aspects of skilled typewriting*. 1983, Springer-Verlag: New York.
4. Fitts, P.M., The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 1954. *47*: p. 381-391.
5. http://www.orin.com/access/SofType/layouts.htm, The Chubon keyboard. 1999.
6. Hunter, M., S. Zhai, and B. Smith. Physics-based graphical keyboard design. in *Proc. of CHI'2000: Human Factors in Computing Systems*. 2000. The Hague, The Netherlands p. 157-158.
7. Karat, C.-M., C. Halverson, D. Horn, and J. Karat. Patterns of entry and correction in large vocabulary continuous speech recognition systems. in *Proc. of CHI'99: ACM Conference on Human Factors in Computing Systems*. 1999. Pittsburgh, PA, USA p. 568-574.
8. MacKenzie, I.S., Fitts' law as a research and design tool in human computer interaction. *Human Computer Interaction*, 1992. *7*: p. 91-139.
9. MacKenzie, I.S., *Spreadsheet calculation of QWERTY and OPTI keyboards*, . 1999.
10. MacKenzie, I.S., A. Sellen, and W. Buxton. A comparison of input devices in elemental pointing and dragging tasks. in *Proc. of CHI'91: ACM Conference on Human Factors in Computing Systems*. 1991. New Orleans, Lousiana p. 161-166.
11. MacKenzie, I.S. and S.X. Zhang. The design and evaluation of a high-performance soft keyboard. in *Proc. of CHI'99: ACM Conference on Human Factors in Computing Systems*. 1999 p. 25-31.
12. Mayzner, M.S. and M.E. Tresselt, Tables of single-letter and digram frequency counts for various word-length and letter-position combinations. *Psychonomic Monograph Supplements*, 1965. *1*(*2*): p. 13-32.
13. McClard, A. and P. Somers. Unleashed, Web tablet integration into the home. in *Proc. of CHI'2000: Human Factors in Computing Systems*. 2000. The Hague, The Netherlands p. 1-8.
14. Norman, D.A. and D. Fisher, Why alphabetic keyboards are not easy to use: Keyboard layout doesn't much matter. *Human Factors*, 1982. *24*(*5*): p. 509-519.
15. Silverberg, M., I.S. MacKenzie, and P. Korhonen. Predicting text entry speed on mobile phones. in *Proc. of CHI'2000: Human Factors in Computing Systems*. 2000. The Hague, Netherlands p. 9-16.
16. Soukoref, W. and I.S. MacKenzie, Theoretical upper and lower bounds on typing speeds using a stylus and keyboard,. *Behaviour & Information Technology*, 1995. *14*: p. 379-379.
17. TextwareSolutions, *The Fitaly one-finger keyboard*, . 1998.
18. Yamada, H., A historical study of typewriters and typing methods: from the position of planing Japanese parallels. *Journal of Information Processing*, 1980. *2*(*4*): p. 175-202.
19. Zhang, S.X., *A high performance soft keyboard for mobile systems*, 1998, The University of Guelph. p. 99.