On-line Personalization of a Touch Screen Based Keyboard

Johan Himberg Jonna Häkkilä Jani Mäntyjärvi

Petri Kangas

Nokia Research Center
P.O. Box 407
00045 NOKIA GROUP, Finland
johan.himberg@hut.fi

Nokia Mobile Phones
Elektroniikkatie 3
90571 OULU, Finland
jonna.hakkila@nokia.com

VTT Electronics
P.O. Box 1100
90571 OULU, Finland
jani.mantyjarvi@vtt.fi

ABSTRACT

The user expectations for usability and personalization along with decreasing size of handheld devices challenge traditional keypad layout design. We have developed a method for on-line adaptation of a touch pad keyboard layout. The method starts from an original layout and monitors the usage of the keyboard by recording and analyzing the keystrokes. An on-line learning algorithm subtly moves the keys according to the spatial distribution of keystrokes. In consequence, the keyboard matches better to the user's physical extensions and grasp of the device, and makes the physical trajectories during typing more comfortable. We present two implementations that apply different vector quantization algorithms to produce an adaptive keyboard with visual on-line feedback. Both qualitative and quantitative results show that the changes in the keyboard are consistent, and related to the user's handedness and hand extensions. The testees found the on-line personalization positive. The method can either be applied for on-line personalization of keyboards or for ergonomics research.

Categories and Subject Descriptors

H.5.2 [Information Interfaces and Presentation]: User Interfaces–*ergonomics, input devices and strategies, theory and methods.*

General Terms

Human Factors, Design, Algorithms, Experimentation.

Keywords

adaptive user interface, virtual keyboard, keyboard design, ergonomics, personalization, handheld devices, adaptation.

1. INTRODUCTION

A mobile phone represents a typical application, where a user has to carry out interaction tasks in a varying environment, with a small device and usually by using only one hand. Furthermore, the situation often requires multitasking, where the attention cannot be focused purely on inputting. The adaptation of the user interface (UI) offers a noteworthy approach to improving the interaction in such environments, as studied e.g. in [1] considering UI adaptation from the base of the user's limited amount of time and working memory for task performing. In demanding situations, improvements in the input method performance are also highly desired. In order to enhance the existing solutions, we have approached this problem by introducing the idea of an adaptive keyboard.

In the human-computer interaction (HCI) research focusing on the UI adaptation, the approach has traditionally been from the application or system point of view. The role of research and implementation of the adaptation of input methods has been in minor role. Partly this is due to the lack of modifiability, since the input methods in HCI have relied of conventional methods, such as mechanical keyboard and mouse. Another aspect includes technological and mechanical limitations that have not allowed possibilities for dynamic modifications. During recent years, novel technical solutions have become more mature and applications in the HCI field more varied both in terms of the devices and operating situations [2]. These factors ease the adapting the user interface from its input point of view.

We have developed a method for adapting the keyboard layout according to its usage by adjusting the position and size of each key. The method monitors the usage of the keyboard by recording and analyzing the keystrokes. The exact locations of keystrokes on the keypad are detected, and the data is used to modify the layout. The layout is gradually changed towards the user's personal way of hitting the keys.

The adapted keyboard layout matches better to the user's physical extensions and grasp of the device thus making the physical trajectories during typing more comfortable. Thus, it benefits the application and product development by allowing a development environment aiming for improvements in the usability ergonomics, decrease in the dialing error rate, and increase in the typing speed.

We believe that this study introduces a novel approach among adaptive user interface research. Previous work has concentrated on either alternative but stable key configurations with improved ergonomic or performance functionality [3,4,5,6], or on-line adaptation where the input events are monitored to prevent typical errors of the particular user when entering text [7].

The adaptive keyboard is a dynamic interaction system but different from applications where modifiability appears with the menu logic, help support, or mistyping in text entry [7,8,9]. It proposes a favorable application of adaptivity that provides subtle and intuitive changes to a conventional layout. Specifically, it does not change the interaction methods or logical/semantic structure of the UI, since the keys in initial layout are presented in an order that follows a familiar convention. This decreases the risk of distracting the users from their habitual use case and comfort factor. Additionally, since the experiment was carried out with a numeric keypad instead of letter entry, there was no need to consider the linguistic factors. These include, e.g., arranging the layout to favor most frequently appearing letters or letter pairs [5,6], or using a conventional layout (e.g., QWERTY) but considering the probability of a particular letter combination when typing has occurred on a boundary of two adjacent keys [10]. Previous settings are neglected in our studies, since number input seldom follow such rules.

In particular, we do not consider Fitt's law (e.g., [11,12]) that is often the fundamental starting point of such studies. Adaptive keypad implies dynamic interaction between the user and the system. To start with, we preferred studying some of its characteristics without involving straight away *a priori* knowledge, such as Fitt's law, in order to optimize its performance.

Consequently, due to the novelty of the idea, the emphasis of this paper is on qualitative results. In order to give ground for further research, we report the basic idea and a particular, experimental implementation of it.

Usability of the experimental implementation of the method was examined with user tests. Verifiable measurement data was collected, and the attitudes of the testees charted in order to guide further development.

We demonstrate and study the applicability of the method both as a tool in for UI research and as a feature in personalizing the end product..

2. METHOD

2.1 Adaptive Keyboard

We start with a general scheme of an adaptive keyboard described in Figure 1.

First of all, the keyboard must be such that the system itself can constantly reform it conveniently, even on-line. Additionally, it must be able to record not only the key that the user presses but also more exact parameters of the touch, in minimum the (x,y) location where the user touches the device. A touch screen is a practical option for this and it is also used in the experiments.

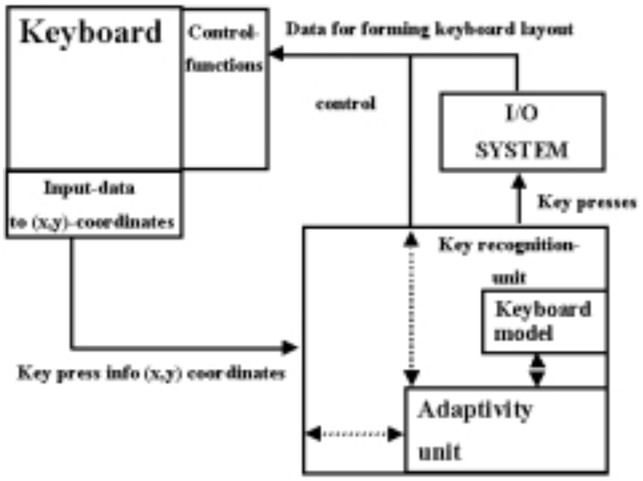

Figure 1. The adaptive keyboard system.

The core of the system is the key recognition unit. Firstly, it recognizes the keystrokes according to the coordinates where the user presses the touch screen. Secondly, it contains a model that determines how the basic I/O system should draw the keyboard. Thirdly, the unit contains an algorithm that can alter the keyboard model based on the information that can be derived from keystroke, i.e., their location, time intervals etc. To achieve this, it can use, e.g., the algorithms that are described later. Additionally, we record whether the key that the user presses is really the intended one, that is, whether the user mistyped something. In our experiments, this is known because of the test arrangements. The system simply dictates the target key the user is to press. In practice, this information can be derived from the corrections that the user makes.

2.2 Algorithms

The following presentation of adaptive algorithms is tailored for a special test keypad, which is designed to simulate a typical mobile phone keypad for dialing. The same algorithms could be extended to some other type of keyboard but we intend to keep the presentation simple.

Specifically, there is no zero key in the keypad to keep the design as simple as possible, that is, to have a rectangular basic layout. In user tests the adaptation always begins from this basic, regular layout (see Figure 6).

2.2.1 Keypad

Each key on the keypad is defined by reference points $\mathbf{c}_i = (c_{i1}, c_{i2})$, $i = 1,\ldots,m$ on the touch screen. In our case, there are $m = 25$ such reference points. Each of the nine keys K_1 = '1', K_2 = '2', …, K_9 = '9' on the keypad are assigned to one key centroid $\mathbf{c}_i$, $i = 1,\ldots,9$ Additionally, there is set of border centroids $\mathbf{c}_i$, $i = 10,\ldots,25$ representing the border of the keyboard. See Figure 2.

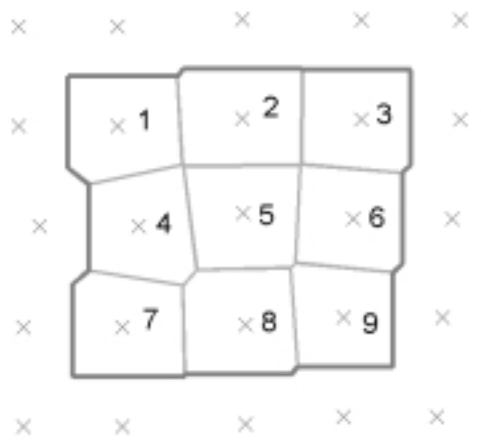

Figure 2. Slightly adapted keypad layout. Labeled crosses (1–9) present the positions of the key centroids c_i, $i = 1,\ldots,9$. The crosses without labels show the border centroids c_i, $i = 10,\ldots,25$. The border centroids are set on a fixed distance b from the neighboring key centroid.

The border centroids are assigned to a null key K_0. The key itself on the visual output is the area of the touch screen that is closer to $\mathbf{c}_i$ than any other reference. The area is marked V_i, formally

$$V_i = \{\mathbf{x} \mid d(\mathbf{x}, \mathbf{c}_i) < d(\mathbf{x}, \mathbf{c}_j), \text{for all } j \neq i\} \qquad (1)$$

where d is Euclidean distance, $i, j = 1,\ldots,m$ and $\mathbf{x}_i = (x_{i1}, x_{i2})$ is any point of the plane. Thus, the keypad is actually a Voronoi tessellation (see e.g., [13]). For the sake of notation, we define the relation from centroids to keys as a lookup table L.

$$L(\mathbf{c}_i) = \begin{cases} K_i, \text{if } i = 1,2,3,\ldots,9 \\ K_0, \text{if } i = 10,11,\ldots,25 \end{cases} \qquad (2)$$

In an adaptation cycle, the user types in a sequence of N key strokes where $t = 1,\ldots,N$. The user is expected to press the target key $K_{\text{target}}(t)$. The actual key $K_{\text{typed}}(t)$ that the user presses is determined in the following way: The coordinates of touch t are $\mathbf{x}_t = (x_{t1}, x_{t2})$. The touch point $\mathbf{x}_t$ is then compared to the key and border centroids. The centroid that is closest to $\mathbf{x}_t$ determines the key (or null key, i.e., the border) that has been pressed

$$K_{\text{typed}}(t) = L(\mathbf{c}_i), \mathbf{c}_i \text{ such that } \mathbf{x}_t \in V_i, i = 1,\ldots,25. \qquad (3)$$

If $K_{\text{typed}}(t) = K_{\text{target}}(t)$ the user has pressed the right key, if not the user has missed the key. Specifically, if $K_{\text{typed}}(t) = K_0$ instead of $K_{\text{target}}(t)$ the user has typed outside the keypad.

2.2.2 Adaptation

The basic idea of the adaptation is that the key centroids are moved towards the centroid of recorded keystrokes in each key. We present two basic adaptation schemes, the batch and the continuous adaptation. The coordinates of the keystrokes are recorded for a longer period, a training cycle $\mathbf{x}_t$, $t = 1,\ldots,N$. Figure 3 shows the original keyboard and the result after one batch adaptation cycle. For continuous adaptation, the key centroid is shifted instantly towards the location of the keystroke (see Figure 4). For batch adaptation, the centroids are moved to the average of keystroke locations in each key (see Figure 3).

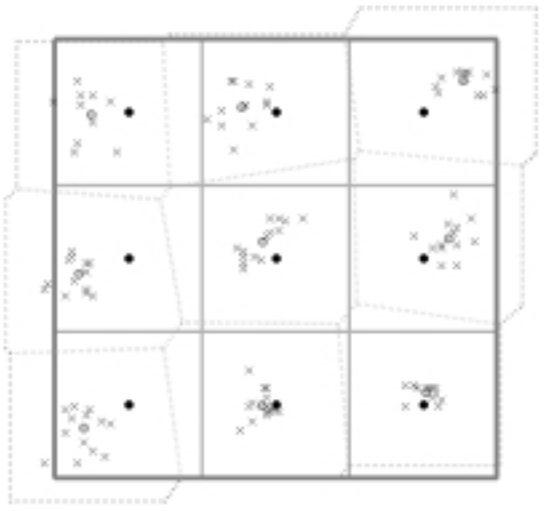

Figure 3. One step of the batch adaptation (from a real user test). Black dots represent original positions of the key centroids and thick lines the borders of the keys. The crosses show the keystrokes on the original keypad. Grey dots and dashed lines represent the centroids and borders after adaptation, respectively. Note that some of the keystrokes have fallen out of the original keypad. They have been discarded from the adaptation.

The keystrokes that are incorrect are discarded in order to avoid erroneous adaptation to unintended keystrokes. In our controlled experiment, the error is known, since the target key that the user should press is dictated by the system. In reality, the same information can be derived from corrective actions, e.g., pressing "backspace".

The steps carried out with batch and continuous adaptation cycles are listed in the following sections 2.2.2.1 and 2.2.2.2, respectively.

2.2.2.1 Batch Adaptation

1. Complete the training cycle and collect $K_{\text{typed}}(t)$, $K_{\text{target}}(t)$, and input coordinates $\mathbf{x}_t$.

2. Find coordinates of correct key strokes for key K_i, i.e., $\mathbf{x}_t$ that fall onto the key K_i, see (1–3) and for which $K_{\text{target}}(t) = K_i$. Let us mark this particular subset of keystrokes $\mathbf{x}^*_i(j)$, $j = 1, \ldots, n_i$ where n_i is the number of such keystrokes.

3. The average of these keystroke coordinates is the new value of the key centroid.

$$\mathbf{c}^{new}{}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} \mathbf{x}_i^*(j) \ . \qquad (4)$$

4. Repeat from step 2 until all actual keys K_i, $i = 1,\ldots,9$ are processed.

5. Set the border centroids on fixed distance from the neighboring key centroids. See Figure 2.

2.2.2.2 Continuous Adaptation

1. Get the input $\mathbf{x}_t$ and find the assigned key centroid vector $\mathbf{c}_i$ and, consequently, the key K_i, see (1–3).

2. If user pressed correct key, the assigned centroid key, $\mathbf{c}_i$, is immediately adapted towards the coordinates of the key press. If the key is incorrect, no adaptation occurs.

$$\mathbf{c}_i^{new} = \begin{cases} \mathbf{c}_i + \alpha(t)(\mathbf{x}_t - \mathbf{c}_i), \text{ if } K_{typed}(t) = K_{target}(t) \\ \mathbf{c}_i, \text{ if } K_{typed}(t) \neq K_{target}(t) \end{cases} \quad (5)$$

where $\alpha(t)$, $0 \le \alpha \le 1$, is a learning rate factor that can be altered during the adaptation.

3. The border centroid(s) is moved if necessary. (See. Figure 2).

Figure 4 presents one step of the algorithm.

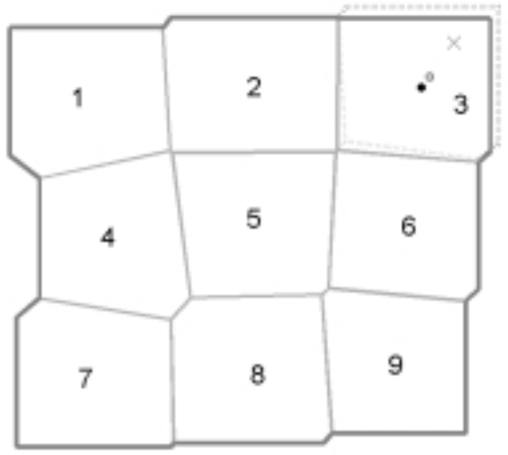

Figure 4. One step of the continuous adaptation (from a real user test). The user has pressed key '3', which adapts. Explanations for the symbols can be found in the caption of Figure 3.

2.2.2.3 Border

The centroids assigned to the border are not directly involved in the adaptation. However, their position is determined by the neighboring key centroid. Keys '2','4','6', and '8' have one associated border centroid and '1','3','7', and '9' three, respectively.

The border centroids are held at the same distance and direction from the neighboring key centroid at all times. For example, for key '1' at coordinates (c_{11},c_{12}) the border centroids are set (c_{11},c_{12})+(0,b), (c_{11},c_{12})+(-b,0), and (c_{11},c_{12})+(-b,b), and for key '6' at coordinates (c_{61},c_{62}) the border centroid is at point (c_{61},c_{62})+(b,0). The constant distance b is the distance between centroids and their closest neighbor on the same row/column in the original layout. See Figure 2.

There exist of course other solutions for moving the border. Another option is to keep the centroids assigned to the border at constant position all the time. This simpler solution leads to problems when the key happens to go very near the border. Consequently, that key grows smaller and may disappear.

2.2.2.4 Background of the algorithms

Both of the adaptation algorithms can be understood as such, as different ways of averaging the input in each key. As one can expect, similar algorithms already exist on other disciplines. For example, both of them can be seen as modifications of basic vector quantization and classification methods found in literature (e.g., [13,14]). The "batch adaptation" is based on performing *one* step of K-means algorithm (e.g., [13]) on keystrokes that were correctly assigned. Consequently, the "continuous adaptation" is related to the learning vector quantization (LVQ) algorithm [14], though, our update rule is simpler in the case of misclassification.

3. EXPERIMENTS

Test equipment consisted of a touch screen laptop PC, which had phone back covers attached to the back of the display for both ends. The input area was located on the edge of the screen. Testees were sitting on front of the office table, where the laptop was placed.

Testing was carried out with twelve testees, most at the age between 25–35 and male. The testees were asked to use their primary hand, which in this particular group meant 3 left-handed and 9 right-handed testees. All testees had to type using their thumb that is a typical way of using hand held devices. See Figure 5.

After the test, general feeling of the adaptive keyboard (in range 1–5) and comments were asked. User's hand was also photocopied in order to record hand extensions. Later an approximate thumb length was measured from these images.

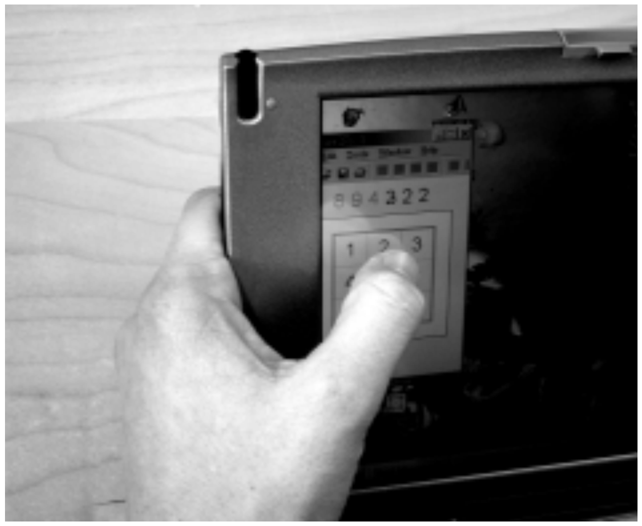

Figure 5. Test situation for a left-handed testee. For a right-handed one the UI is moved to the right end of the display.

The UI including the original keypad layout is presented in Figure 6. The dimensions of each key were 10mm×10mm. The dimensions were chosen in order to compromise the realistic device size and human performance. A bounding box was drawn at 5mm distance around the keypad to imitate an environment where the input detection area is limited, as it is in real applications. The box formed just a visual guidance for the keyboard area, but did not limit the adaptation of the keyboard.

The number sequence of the target keys to be dialed was placed above the keyboard so that the visual focus point did not have to be changed much between the sequence and keyboard, and the dialing could be executed by glancing occasionally, if required.

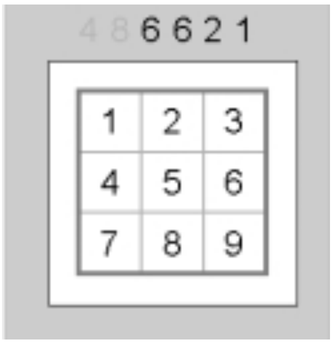

Figure 6. The original appearance of the user interface. The user has correctly fed in '4' and '8'.

Visual feedback was provided to the user in form of changing the colour of the numbers in the sequence as the dial was executed. Right key stroke was indicated by green and mistype by red color. Feedback of a mistype was added to the test after a pilot testing phase along the wishes of the pilot testees, who found the lack of it confusing. There was no possibility to correct the mistypes, however.

The target dialing number consisted always of six random digits 1−9. They were presented to the testee as cycles of either 12 or 21 six-digit sequences resulting in 72 or 126 digits per cycle. The choice of the cycle length was partly dictated by the need to have exactly the same number of each digit in a cycle. Otherwise the unbalanced distribution of digits might guide the keypad adaptation. The target cycle was constructed by shuffling appropriate number of digit series 1, 2,…,9. The testing consisted of two parts, and a short training session was provided to introduce the test.

Table 1: Test series

Cycle	*Layout in the beginning of the sequence*	*Adaptation*	*Number of sequences*
T1	Training cycle for batch adaptation (12 sequences)		
B1	Original	batch adaptation after completing the sequence	21
B2	result of B1	like B1	12
B3	result of B2	like B1	12
B4	result of B3	like B1	12
B5	result of B4	like B1	21
T2	Training cycle for continuous adaptation (12 sequences)		
A1	Original	no adaptation	21
A2	(original)	continuous adaptation	21
A3	result of A2	no adaptation	21

Batch adaptation was used in the first part of the test. It consisted of five dialing cycles and four batch adaptations in between. We recall that the keypad appearance remained the same during the dialing cycle. The three intermediate dialing cycles were shorter in order to speed up the adaptation process, and reduce the fatigue of the testee. The first and last cycles were longer in order to provide more stable statistics for calculating the typing speed for original and adapted keypad.

The second part of the test used continuous adaptation. In this case the test series was simpler since there is no division between input and adaptation. The first cycle was for measuring the performance on the original keypad, the second one for the adaptation, and the last one for measuring the performance on the adapted keypad. The test series is described in detail in Table 1. The acronyms for the test cycles are used in Results.

4. RESULTS

4.1 Keypad shape, size, and position

Test results show consistent changes in keyboard layout. Firstly, the size of the keyboard layout tends to grow along with the adaptation. The keypad area in B5 was 1.4–1.9 (lower–upper quartile) times the area of the original one. The same holds for continuous adaptation when comparing the original one to the end result of A2 whose area was 1.2–1.7 times bigger. See Figure 7.

Secondly, the keyboard size correlates slightly with the testee’s thumb length. The correlation coefficients between the square root of the adapted keypad area (“the keyboard radius”) and the testee thumb length were calculated for the test cycles. In batch test cycles B2, B3, B4, and B5 the linear correlation coefficients are +0.42, +0.51, +0.51, and +0.50, respectively. For the end result of A2 the correlation coefficient is +0.35.

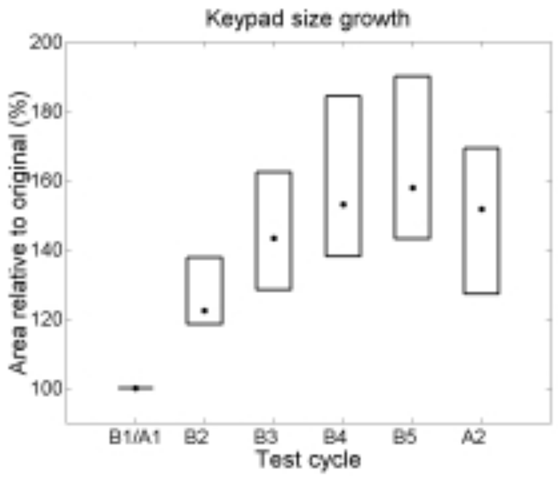

Figure 7. The size of keyboard layout along with adaptation.

In general, with right-handed testees, the center of the keyboard is found to move towards left. See Figure 8. However, there are few exceptions. For some testees, the center moves also clearly upwards, and for one clearly downward as shown by Figures 8 and 12. At least partly, the reason for this may be that the test equipment design did not especially guide the vertical level of the user's grasp. These users might have kept their hand on a considerably different level than the average testee. This has then moved the keypad up or down.

Thirdly, the resulting keypad layouts significantly resemble each other in shape. The shape of the keyboard adapts due to the natural thumb muscle trajectories so that near upper corner stretches up and towards the palm and distant and lower corner extends. This can be seen in the averaged final adaptation pattern for left- and right-handed testees in Figure 9. Figures 9 (left) and 9 (right) seem like mirrored versions of the somewhat similar pattern. Obviously, the result is not as clear for left-handed testees resulting from only three samples. Figures 11–12 present two individual cases of this pattern.

4.2 User feedback and dialing performance

When estimating the performance of the adaptive keyboard, we have to rely mainly on testees' comments. We recall that the test was not designed to measure performance. There are two main reasons that prevented us from making firm conclusions from the dialing speed statistics in Figure 10. Firstly, the user latently learns to type faster in general during the test. This is clearly seen when comparing the identical cycles B1 and A1 in Figure 10. In B1 and A1 the user types into the original keypad.

Secondly, decreasing dialing performance in the long run may be due to fatigue and demotivation. The lack of tactile feedback has reported to increase the force that is used for dialing [15]. This is clearly a drawback when using an ordinary touch screen. Fatigue and tension in finger muscles is one relevant factor that decreases the dialing speed at the latter part of the test series, which was also pointed out by some of the testees.

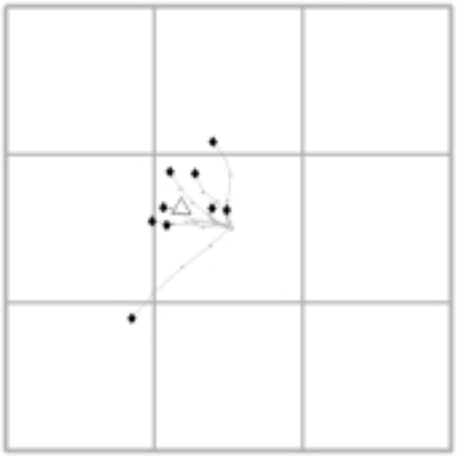

Figure 8. Evolution of the position of the center of mass of all keys for right-handed testees in cycles B1–B5. Thin gray line shows the trajectory B1–B5, black dot indicates the final position (in B5). The triangle shows the average of final center positions. The original keypad layout is shown here for reference. Final keypad B5 for the most extreme shift (at southwest) is show in Figure 12.

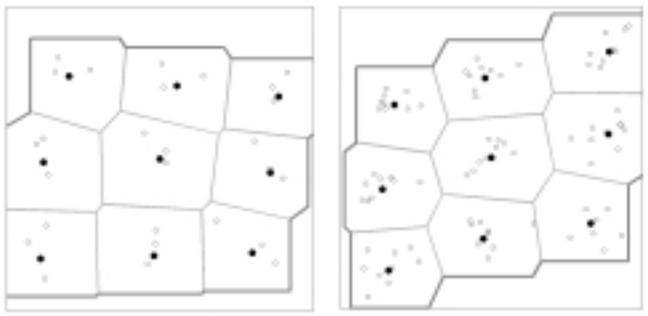

Figure 9. Average keypad shape after cycle B4. Left and right panels show the average keypad for left handed and right handed testees, respectively. The open circles show the key centroids of individual testees. The black dots according to which the keypad is drawn are averages of these.

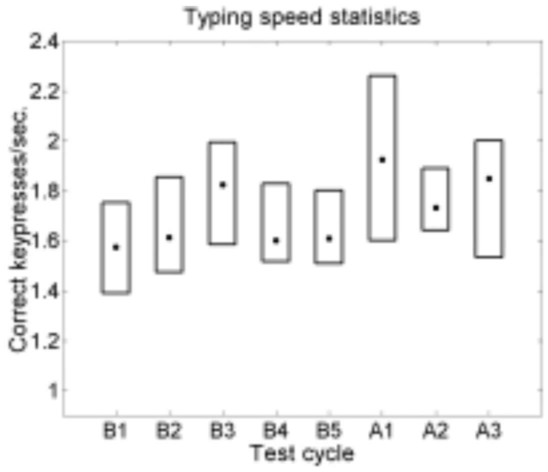

Figure 10. Correct keystrokes per second in the test cycle. The box includes upper and lower quartiles of the testees' typing speed for the cycle. The dot indicates median.

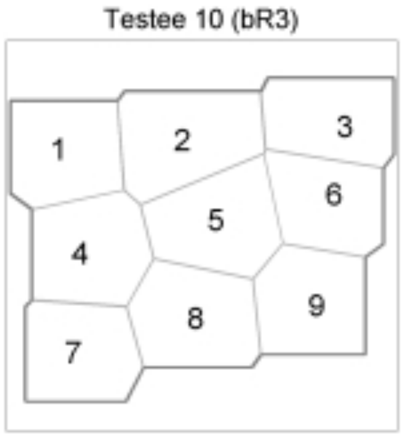

Figure 11. Typical result.

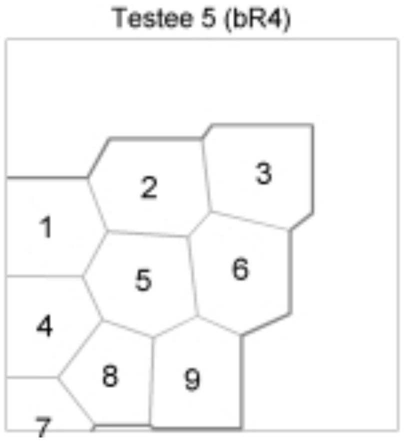

Figure 12. Shifted keyboard. The overall form is typical, but the center of mass has moved.

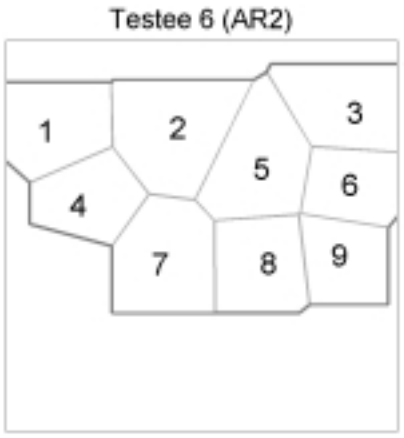

Figure 13. Untypical shape from continuous adaptation for a testee. This keypad is a result of disturbingly abrupt changes that may occur during fast adaptation. See text for discussion.

The average feeling about the adaptive keyboard reported on scale 1–5 was 3.8 (σ= 1.1), i.e. positive. A typical comment, given by 5 of the 12 testees, was the feeling that the keyboard adapted too much. The general opinion was that a comfortable stage was reached in the earlier adaptation stage (B2–B3). The highest dialing speed in the batch test was also attained in the same cycles. It appears, that the recorded dialing speed in test cycles B1–B5 is in accordance with the reported comfortable stage and over adapting. But as we already pointed out, any affirmative conclusions cannot be made from typing speed statistics alone.

The batch adaptation in B1–B5 was generally more appealing than the continuous adaptation in A1–A3. Again, this accords decreasing dialing speeds in A2 and A3 compared to A1 in Figure 10.

It was mentioned that the disorganized layout pattern was found distracting. This happens since the boundaries of each key were shown, and the layout formed rather indented pattern when the adaptation advanced.

Since the keypad typically grew larger, some keys partly escaped the active area. This happened in the later phase of the adaptation process.

5. DISCUSSION

The results show that the adaptation produces a keypad shape that may be regarded reasonable with respect to handedness and natural thumb muscle trajectories. This is a good result, compared to the simplistic adaptation algorithm. Each key is parameterized with just one center point. In addition, the adaptation process does not take into account other factors than touch center point. For example, area of touch, time between keystrokes and keystroke distribution within a key are not explicitly modeled in the adaptation algorithm.

A detail that may be criticized in our test arrangements is the dictated dialing sequence. The main reason for having predetermined user input was to ensure that the distribution of the dialed numbers was uniform. This was necessary in order to avoid an additional, uncontrolled factor in the test: The pilot experiments clearly indicated that if the dialing sequences consisted of only some numbers, say, the diagonal ‘1’,’5’, and ‘9’, the resulting keyboard layout would be clearly different from the result that was achieved by using all numbers evenly. This is a fact that is intuitively thinking quite evident. Since we knew the targets, we also could eliminate them from the adaptation. If we think of our particular adaptation algorithm itself, it really should be tested separately whether or not eliminating the “unintentional” or “wrong” keystrokes is really crucial. Taking into account the averaging that happens during the adaptation, it might even be possible that the results would not change much after all even if the “typos” were let to adapt the keyboard. However, the dictated input target gave us, as a byproduct, a simple way of controlling this factor, without specifically incorporating a system that derives this information from the corrections done by the testee (pressing backspace etc.).

In general, the changes in physical layout of the adaptive keyboard are intuitive (see Figure 3) and often rather small. Moreover, the application does not require any changes in interaction methods. The application supports the feeling that the user is having the control over the device, which makes the interaction process more accessible. Thus, it avoids the pitfall of the user being distracted by feeling of a lack of control over the device, i.e. application behaves unexpectedly [16].

Occasionally, the keyboard pattern became misleading in its presentation of information. This may happen if the individual keys change places, e.g., by wedging a key to the wrong row like in Figure 13. Big changes in the user interface introduce visual disturbance that confuses or even irritates the user. However, the user’s mental model is based on conventional and widely used layout, like the 3×3 keyboard. The conflict between the mental model and visual appearance of the keyboard may also arouse the feeling of discomfort during adaptation [17]. Thus, it affects the keyboard usability, speed and error rate of dialing. In our case, this effect was probably most noticeable in continuous adaptation. The adaptation speed was set quite high in order to keep the length of the test moderate and still achieve notable changes. This may have introduced a feeling of unstableness, small unfortunate changes in the beginning may have directed the adaptation into an unwanted direction and resulted into a pathological layout that Figure 12 exemplifies. Moreover, the continuously changing layout may have contributed to dislike and low typing speed of A2.

As this was a first experiment, the number of variables to be changed was limited. The keyboard layout, the key shape parametrization and the adaptation algorithm were designed to be simple.

Various experiments could be done with more a sophisticated arrangement, where the impact of, e.g., alternation in physical dimensions and design ergonomics could be investigated more thoroughly. Also by enabling the measurements of the finger touch surface and the dialing force would offer interesting research viewpoints.

In order to ensure comfortable usage, adaptation should be carried out so that the changes do not distract the user. To avoid that, the borders of each key could be made transparent, or the keys appear smaller than the actual detection area. The keys could also have a more regular shape, e.g., a circle or an oval. The keypad adaptation should also be bound by the physical limitations of the device. In our case, this would mean preventing the keys from escaping the active area. On the other hand, it is an interesting research question if an adaptive keyboard with unlimited size would reach a stable configuration using a certain algorithm.

The methods can be developed in two directions. Firstly, one can use the adaptation in HCI research. Secondly, the adaptation can be used as part of the personalization of a handheld device keyboard.

In order to ease the use and increase the performance level of the keyboard, the system should be able to receive feedback to monitor and adapt itself towards a more effective layout. By taking into account, e.g., the error dialing frequencies, relations or over adapting sizes of distinct keys, the adaptation algorithm could be modified to produce a better performing keyboard.

The primary results show that the adaptive keyboard approach offers a usable tool to investigate and design keyboard ergonomics.

6. ACKNOWLEDGMENTS

We gratefully acknowledge Mr. Miika Silfverberg and Mr. J. A. Flanagan for their valuable comments.

7. REFERENCES

[1] A. Jameson, R. Schäfer, T. Weis, A. Berthold and T. Weyrath, "Making Systems Sensitive to the User's Time and Working Memory Constraints", in Proc. 1999 Intelligent User Interface Conf., Redondo Beach, CA, 1999, pp. 79–86.

[2] J. Mäntyjärvi, J. Koivumaki and P. Vuori, "Keystroke Recognition for Virtual Keyboard", in Proceedings of the IEEE International Conference on Multimedia and Expo, Vol.2, pp. 429 -432, 2002.

[3] M. Silfverberg, I. S. MacKenzie and P. Korhonen "Predicting Text Entry Speed on Mobile Phones", in Proc. of the CHI 2000 conference on Human factors in computing systems, pp. 9–16, ACM Press, 2002.

[4] G. W. Lesher, B. J. Moulton and D. J. Higginbotham, "Optimal Character Arrangements for Ambiguous Keyboards," IEEE Transactions on Rehabilitation Engineering, pp. 415–423, 1998.

[5] I. S. MacKenzie and S. X. Zhang, "The Design and Evaluation of a High-Performance Soft Keyboard" in CHI 1999, Pittsburgh, PA, 1999, pp. 25-31.

[6] S. Zhai, M. Hunter and B. A. Smith, "The Metropolis Keyboard – An Exploration of Quantitative Techniques for Virtual Keyboard Design", in Proceedings of ACM Symposium on User Interface Software and Technology 2000, San Diego, CA, pp. 119-128, ACM Press, 2000.

[7] S. Trewin and H. Pain, "A model of Keyboard Configuration Requirements", in Proceedings of the Third International ACM Conference on Assistive Technologies, pp. 173-181, ACM Press, 1998.

[8] H. Maeda, K. Haro and N. Ikoma, "An Intelligent Database Interface Adapting Itself to the Degree of User's Skill" in 1999 IEEE International Fuzzy System Conference Proceedings, pp. 1153–1158, Aug. 1999.

[9] R. Trevellyan and D. P. Browne, "A Self-Regulating Adaptive System", pp. 103-107. CHI+GI 1987.

[10] J. Goodman, G. Venolia, K. Steury and C. Parker, "Language Modeling for Soft Keyboard" in Proceedings of Intelligent User Interface '02, San Francisco, CA, pp. 194-195, 2002.

[11] N. Walker and J. B. Smelcer, "A Comparison of Selection Times from Walking and Pull-Down Menus", in Proc. of CHI '90, pp. 221-225, 1990.

[12] I. S. MacKenzie and W. Buxton, "Extending Fitts' Law to Two-Dimensional Tasks" in Proc. of CHI '92, pp. 219-226, 1992.

[13] A. Gersho and R. M. Grey, Vector Quantization and Signal Compression, in Kluwer International Series in Engineering and Computer Science, 159. Kluwer, 1992.

[14] T. Kohonen, Self-Organizing Maps, in Springer Series in Information Sciences, 30. Berlin Heidelberg: Springer-Verlag, 1995.

[15] D. A. Thompson, "Keyboard Tactile Feedback and its Effect on Keying Force Applied" in Human-Computer Interaction, Vol. 1, H.-J. Bullinger and J. Ziegler, Ed. 1999, pp. 157–161.

[16] B. Shneiderman, Designing the User Interface, 3rd ed., Addison-Wesley, 1998, pp. 86–87.

[17] R. Spence, Information Visualization. ACM Press / Addison-Wesley 2001, Ch. 6.