Comparison of Two Touchpad-Based Methods for Numeric Entry

Poika Isokoski and Mika Käki
Department of Computer and Information Sciences
FIN-33014 University of Tampere
Finland
poika,mk@mail.cs.uta.fi

ABSTRACT

Small hand-held touchpads can be used to replace stylus-based digitizing tablets when the use of a stylus is not convenient. In text entry tasks where the writing surface is held in a hand the error rate becomes a problem. The small size of strokes compared to the width of the fingertip and the additional imprecision caused by the interaction of the pad and finger movements make input very imprecise. We describe a new improved clock-face based stroke system for entering numbers with a touchpad. In a 20-session user study with 6 users we found slightly better throughput of successfully entered numbers with the proposed new system. This advantage was mainly due to lower error rate with the new system. User preference similarly slightly favored the new system over an earlier straightforward proposal based on the clock metaphor.

Keywords
Clock metaphor, writing, mobile devices, stylus overhead

INTRODUCTION

A whole family of computing devices has recently sprung up in the size range between mobile phones and laptop computers. These devices, known as Personal Digital Assistants (PDAs) are with few exceptions pen-operated because keyboards would be impractical. The display area of the device usually reacts to touch by a finger, but a special stylus is often provided for more accurate pointing.

A pen-based input incurs a certain cost. The user is required to find the stylus, take it into his or her hand, and put the stylus back once the session with the PDA has ended. With PDAs in the usual calendar, notebook, and messaging contexts this cost in time and effort has proven to be tolerable. However, if the task and device at hand are sufficiently simple such as changing the channel on a TV set or changing settings on a digital camera or a wristwatch, the stylus overhead may be too high. Furthermore, some devices are too small to have a good storage place for the stylus making losing it alarmingly easy. We explore the possibility to forego the stylus and use the more imprecise technology of finger touch sensing.

We compare two methods for the entry of numbers using a handheld touchpad. The first method is a re-implementation of a system used in an earlier study by McQueen *et al.* [17]. It is based on the idea that a character is a stroke on an imagined clock-face from center towards the desired number. This system is a straightforward implementation of the clock-face metaphor and therefore called the *pure* clock-face design in this paper. The second method is new and uses roughly the same principle, except that off-axis strokes are L-shaped following first the nearest axis and then turning towards the desired number in the clock-face (see Figure 1). This second system being a mixture of the pure clock-face design and other systems is hereafter known as the *hybrid* design.

Previous work in the area suggests that the error rate with the pure design is disturbingly high. We expect the error rate to further increase when using a handheld touchpad and finger instead of a stationary tablet and a stylus. The hybrid system could be expected to have poorer overall performance because the more complex L-shaped strokes may be slower to draw. However, our hypothesis is that the hybrid stroke system is more robust and therefore has lower error rate allowing comparable or better overall performance.

In the remainder of this paper, we explore this hypothesis by first explaining previous work that inspired us to experiment with this kind of number entry systems. Then we give details on our prototype design and implementation followed by a description of an experiment and results relating to the hypothesis. We finish with conclusions based on the results of the experiment.

RELATED WORK

The present work is closely related to two pieces of earlier research. The first set of related papers describes experiments with a numeric entry technique based on a clock metaphor [15, 17]. The second set of closely related papers is the work on marking menus [12, 13, 14].

Additionally, the work with touchpad based remote-controllers [8] and various unistroke alphabets [1, 6, 9, 11, 16, 18, 19] is highly relevant to the work at hand. Especially so, when related to the notion of sloppiness space in the design of unistroke character sets as discussed by Goldberg and Richardson [9].

We will now discuss in more detail the two areas of research that led us to the present work.

Marking Menus

Pie menus are menus that are drawn by dividing a circle into sectors. Selections are accomplished by starting from the center and moving the pointer over the desired slice and clicking (or releasing) a button on the pointing device. Because the pointer must be in the center of a pie menu when the selection begins, pie menus are most conveniently used as context sensitive menus that pop up wherever the cursor happens to be. Just like the more common linear menus, pie menus can be nested. When a slice representing a submenu is selected, the submenu is shown around the cursor. The chain of menu selections thus continues until it is interrupted or a leaf node in the menu structure is reached.

Marking menus are special kind of pie menus. In addition to regular pie menu behavior they allow selections by using the pointer movements without showing the menu. When the marking behavior of the menus is used, the pointer is used to draw the path that would select the desired slice if the (appropriately scaled) menu were visible. If the pointer does not move, the menu is shown after short time delay to make selecting easier for novices.

The usefulness of pie menus in general [5] and marking menus more specifically [12, 14] is well established by now. Pie and marking menus are faster to use than linear menus in many graphical user interfaces. The connection to text entry is easy to draw given that some experiments aiming to evaluate marking menus have also been experiments in text entry with a limited character set (For an example of such an experiment see [13]).

Kurtenbach and Buxton explored user performance on marking menus with varying widths and depths [13]. Their results show, among other things, that selection times on a one-level 12-slice menu are close to those measured on a two-level 4-slice menu. On the other hand the error rate on a two-level 4-slice menu is only about half of what was measured for the one-level 12-slice menu. This is the critical finding that we try to verify and utilize in the present work.

Menu systems specifically aimed for text entry have also been proposed. T-Cube by Venolia and Neiberg uses an initially visible 8-slice circular menu [18]. A touch on one of the sectors initiates behavior very similar to standard marking menus: The user may draw the gesture directly, or wait for a 8-slice sub-menu to appear. In T-Cube the submenu does not appear under the pen but elsewhere to avoid being obscured by the pen and the hand. Other systems like Quikwriting [18] and Minimal Device Independent Text Input Method (MDITIM) [11] can also be seen as derivatives of menu selection techniques or more generally tree traversal techniques. However, these systems do not show the menu to the user and for the most of the time the user's impression is not that he or she is using a menu.

For the remainder of this paper we will talk about the strokes used for number entry as if they were handwriting characters like any other unistroke characters such as the original unistrokes by Goldberg and Richardson [8], Graffiti [1], or MDITIM [11]. However, the alternative interpretation as menu selection traces would be equally valid.

Number Entry

Systems capable of general text entry can certainly be used for entering numbers too. The difficulty of learning the character set (or number locations in the menu) should be alleviated to allow good novice performance. A central idea in marking menus is that the menu provides automatic context sensitive help that aids in the learning of the gestures.

For entering numbers, familiarity with the number arrangement on a clock-face can be used as a memory shortcut for new characters. McQueen *et al.* compared a system based on the clock metaphor (Pie pad) and regular hand written characters for entering numbers [17]. Pie pad strokes are the same as the "pure" system in Figure 1. For the regular handwriting part they used the Microsoft's handwriting recognizer version 1.0. The regular handwriting part of their study, however, is not very interesting in the context of the work at hand. The clock metaphor part of the study is what we wish to elaborate on. McQueen *et al.* had six students use both systems for 20 sessions, each session lasting 15 minutes per system. They found that the clock metaphor based system is initially slightly slower, but with practice soon overtakes regular handwriting in speed. Error rates did not differ significantly. During the last three sessions the average error rates were close to 8% for both techniques.

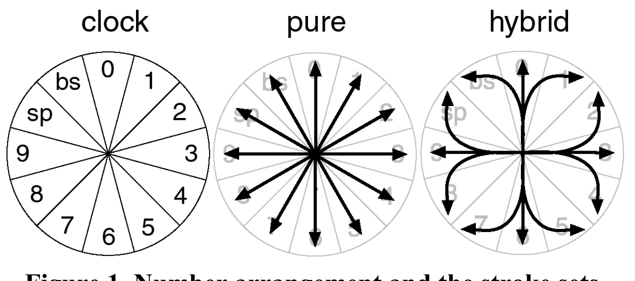


Figure 1. Number arrangement and the stroke sets.

The system that we call the pure clock-face system in this paper is essentially the same as PiePad. The two main differences between our system and the one used by

McQueen *et al.* are related to the differences in the used pointing technology. First, we use a handheld touchpad instead of a fixed stylus operated pad. Second, the point where the text is drawn with the finger is dissociated from the point where the text is entered. In other words we use an indirect pointing device whereas McQueen *et al.* used direct pointing.

INTERFACE DESIGN AND IMPLEMENTATION

Method 1: Pure Clock-Face

Design

We copied the design of the system used by McQueen *et al.* as closely as we could. A schematic of the character set is shown in Figure 1. Although strokes for 10 (space) and 11 (backspace) do exist, they were not used in our experiment.

The strokes were recognized based on the first and last points belonging to the stroke. Only the direction of a vector drawn from the beginning of a stroke to the end was used for recognizing the stroke. We chose to ignore the suggestions for improvement of the method given by McQueen *et al.* because we wanted to be able to directly compare our results on speed and error rate with their results.

Implementation

As an input device we used a Cirque EasyCat touchpad shown below in Figure 2. The device is intended for desktop use, but is small enough (69x86x13mm) to be held in a hand. The touch-sensitive area available for use is approximately 45x60mm. All of this area is not needed for this method. An area of about 20x20mm should be enough, although it should be noted that decreasing the size of the strokes increases error rate. This happens because the noise in finger movements begins to interfere with the intended shape of the stroke. Especially the relatively frequent hook-shaped appendices in the beginnings and ends of the strokes tend to cause problems when overall stroke size is small enough. In our implementation, we did not limit the area use in any way.

We needed the touchpad to produce high-resolution absolute position data with pressure information. Therefore, we opted to use Shaun Bangay's Cirque Cat Driver for Linux version 0.1.8 [4] as a basis for our own C++ class that handles the touchpad interface over the serial port.

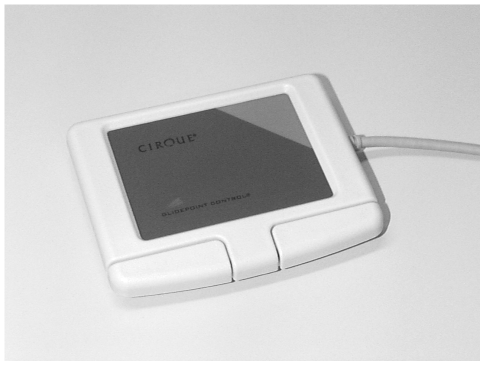


Figure 2. Cirque EasyCat touchpad.

Our interface class spawns a Linux thread that reads the data from the touchpad and maintains a time-stamped event queue that can be read later without loss of accuracy in the timing of the events. The test software reads events from this queue and draws the finger trace in the feedback window (Figure 3). When the end of the stroke is received, its characteristics are computed and number recognition takes place. The number is then forwarded to the task window (seen in Figure 4). The whole process is fast to the extent that the system appears to respond in real time. Possible differences in running times of the recognition algorithms are in a scale smaller than the refresh interval of the display (11,8 ms) and are therefore unlikely to contribute significantly to the results of the experiment.

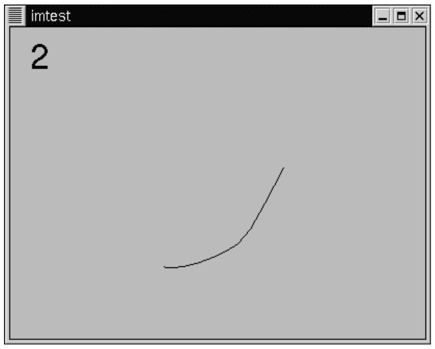


Figure 3. The feedback window.

Method 2: Hybrid Clock-Face

Design

This method for number entry is our own design. Parts of the system have been described before, but the combination is new. The main problem that the design tries to solve is the difficulty in consistently drawing lines that end within one of the 30 degree slices in the pure clock-face design. Especially the off-axis directions (1, 2, 4, 5, 7, and 8) are more error prone than is desirable [13, 17]. The solution is to encode the off-axis directions into sequences of on-axis strokes. For example, stroke for number 1 is combination of strokes for 0 and 3. On axis strokes 0, 3, 6, and 9 are written just like in the pure design (see Figure 1). The important feature of this design is that users can still use the clock metaphor to remember where the stroke should end.

Implementation

The hardware and software used for the hybrid design are the same as described above for the pure clock-face design except for the stroke recognizer. Our pure clock-face recognizer would recognize correctly the hybrid strokes shown in Figure 1. However, because it is unlikely that people will be able to write the curved strokes so that they end so neatly within the correct slice, we used a different kind of a recognizer. Our recognizer is a slightly simplified

C++ rewrite of the recognizer that was used by Enns and MacKenzie in their work on touchpad-based remote control [8].

In essence the recognizer is a very simple greedy exact matching feature extractor kind of a recognizer. Greediness here meaning that it returns the first instance in the dictionary that matches the feature vector extracted. Exact matching means that only an exact match is returned instead of a list of the best matches as is commonly done in more advanced recognizers. The features needed for recognizing the characters used in the experiment include the first, second, and third quadrant of a 2x2 grid (bounding box divided in four) that the stroke enters, and the normalized length of the stroke. The length was computed in three different ways. First, only considering the movement along the x-axis of the touchpad coordinate space. Second, the movement along the y-axis, and finally using the two-dimensional distances between consequent points within the stroke.

As a consequence of the features of the recognizer, the slices in the menu interpretation of the characters are not 90 degrees wide. The straight strokes (0, 3, 6, and 9) have slice width of 30 degrees. The first part of the curved strokes has an effective slice width of 60 degrees. The effective width of the second part of the curved strokes depends on the direction of the first part and the relative lengths of the parts. In all cases, however, the effective width for the second slice is greater or equal to 30 degrees.

EVALUATION

Method

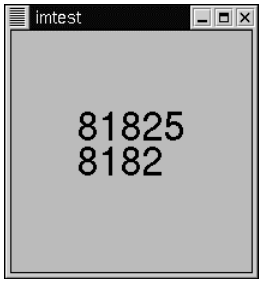
Participants

6 people, all students or staff of the University of Tampere, volunteered to serve as unpaid test users in our experiment. Ages of the users varied between 21 and 30 years. Three of the users were male and three were female. None of them had used a touchpad for extended periods of time before the experiment. Three users had owned and used an analog wristwatch for several years and three had not. All users were right handed.

Apparatus

The participants sat in front of a regular office workstation. They held the touchpad in their non-dominant hand and used the index finger of the dominant hand to draw the characters.

The computer display was positioned on the desk in front of the subjects. There were two windows visible on the display. The first of these was called the feedback window (shown in Figure 3). It showed the last stroke drawn or the portion of the current stroke drawn so far. After the user lifted his or her finger, the character was recognized and shown on the upper left corner of the feedback window. In case of the pure recognizer, a very short stroke (under 2 mm) was rejected and "too short" appeared instead of the recognized character. Also, if the recognizer in the hybrid condition failed to recognize the character "unrecognized" was written. The second window is the task window shown in Figure 4. It is our approximation of the kind of task that McQueen *et al.* used in their experiment with PiePad. The upper string of five numbers seen in Figure 4 is the string to be entered. Correctly entered numbers appear below this string. Incorrect numbers were not accepted, but were instead logged as errors. A “click” sound was played when the task window received a character. The same sound was played regardless of whether the character was correct or not.



81825
8182

Figure 4. The task window.

Procedure

Each test subject completed 20 sessions. There were at least three hours, but no more than three days between two consecutive sessions. Each session consisted of two 15-minute subsessions: one for the pure clock-face design and one for the hybrid design. The order of the subsessions was switched for each session. Each session ended with a question on which of the two systems seemed better during that session. With some setup time before each subsession and short rest between the subsessions, each session lasted approximately 35 minutes.

The first and last sessions were slightly longer. The first session began with a short interview to collect the user demographics. After that the experimental task was introduced. Next, a 5-minute session using the numeric keypad on the keyboard was completed. After that first session proceeded like all other sessions. On the last session the exceptional activities were at the end. After completing the regular 15-minute subsessions with the touchpad, the 5-minute keyboard session was repeated. The last session ended with a short interview, during which the users were given an opportunity to express their opinion on the experiment. The background and the goals of the experiment were also explained to those who wanted to know.

Within each subsession a new sequence of five numbers was presented immediately after the user completed the previous one. Thus, the users were required to keep entering the numbers continuously with no breaks. Because the users were instructed to work as fast and as few errors as possible, we hoped that the degree to which the number

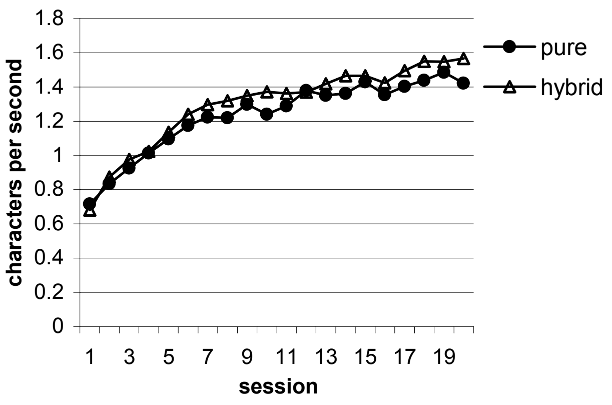


Figure 5. Throughput in characters per second.

entry methods cause physical or mental fatigue would show in the overall throughput and error rate.

Design

The experimental design for comparing the two methods was within subject factorial design with the writing method (pure, hybrid) and amount of practice as the independent factors. Entry speed (throughput and timing of the strokes) and error rate were measured as dependent variables.

Results

Throughput

The most interesting measure of performance for text entry systems is the rate at which users can get their work done despite the errors and other distractions that happen. As an approximation of this measure we computed the average number of correctly entered numbers per second over all the users for each session. The result of this computation is shown above in Figure 5. In addition to the obvious main effect of session a repeated measures ANOVA (used to compute all the F values reported in this paper) does not show significant main effects or interactions. Most importantly the main effect of the system is not significant ($F_{1,5}$=4.48,p>0.05). Although, this effect is not statistically significant, we expect it to be real due to reasons given later in this section.

In comparison to the standard keyboard numeric keypad that represented the normal way of inputting numbers in our experiment, both touchpad methods are slightly slower. However, as indicated by the large overlap of the error bars in Figure 6, the differences are not significant. The number for the keyboard is the average of the two 5-minute sessions. The numbers for the touchpad-based methods are averages over the last three sessions.

Timing

For easy comparison with McQueen *et al.*, we measured the time spent writing the characters using the same sub-categories. That is, in addition to the overall time used per character, we measured the time that the finger was in contact with the touchpad (scripting time) and the time spent between the contacts (preparation time). The average

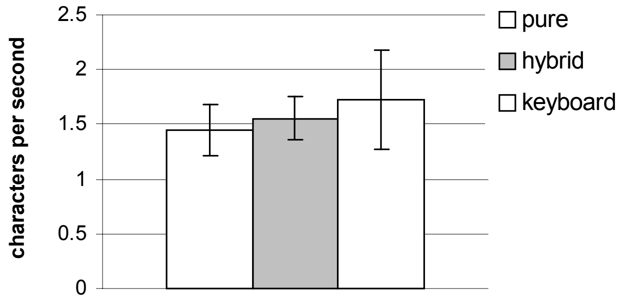


Figure 6. Average throughput for the three systems.

total time spent per correctly entered character is shown in Figure 7 (the upper two series labeled total). Neither of the two systems seems to be clearly faster than the other ($F_{1,5}$=0.049,p>0.05). The split into preparation and scripting time reveals a difference between the systems. The data in Figure 7 suggests that the use of the hybrid method leads to longer scripting times ($F_{1,5}$=37.2, p<0.05). On the other hand preparation times for the hybrid method seem to be shorter ($F_{1,5}$=5.56, p<0.1). These differences cancel each other out resulting in roughly equal overall entry time. This does not match the observed relationship between the throughput means in Figure 5. To understand why the throughput curves differ from the mean entry time curves for successfully entered characters we must consider the effect that the unsuccessfully entered characters have on the throughput. In other words, we need to take a look at the error rates.

Error rate

Only correct input would allow the user to advance in the task that we used in our experiment. All erroneous input was therefore done in addition to the correct input. Error rates reported below are computed with the formula:

$$error\ rate = \frac{incorrect}{correct + incorrect}$$

so that if there is no correct input at all, the error rate is 1 (or 100%) and if there is no incorrect input at all, the error rate is 0.

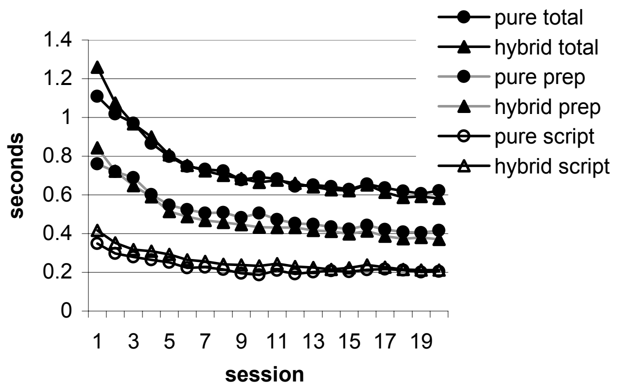


Figure 7. Time spent per correct character.

There are two kinds of excess input. First, there are strokes that were not recognized. These include the too short ones for the pure clock-face condition and the strokes of unrecognizable shape in the hybrid condition. Second, there are characters that were recognized, but that were wrong, that is they did not match the sequence in the model string.

First we will take a look at the total error rate that includes both error types. In Figure 8 we have the mean error rates for both systems over all the sessions. The error bars show the standard error of mean for each system and session. A repeated measures ANOVA shows a significant main effect for the system ($F_{1,5}$=45.6,p<0.01), but no other significant effects. Making more errors leaves less time for productive work. Therefore, this significant difference in error rate substantiates our earlier claim for real difference in average throughput.

During the experiment it became obvious that some characters were more difficult to enter than others. Figure 9 is an illustration of this. The data points are located on the centers of the sectors in the clock-face design. The distance from the center of the plot gives the error rate for the character in question. The profile for the pure clock-face design is similar to what McQueen *et al.* reported for PiePad. The off-axis strokes were more error prone. In the hybrid profile only strokes for 4, 5, and 7 show this tendency.

Although the hybrid clock-face method exhibits lower overall error rate computed over all the sessions (10.2% for hybrid vs. 14.6% for pure), there are further differences in the behavior of the methods. Namely the errors are distributed differently into the two different error types. Figure 10 shows the different error types. The points labeled "wrong" represent strokes that were recognized, but resulted in input that was wrong at that point of time. The points labeled "unrecog" represent strokes that were not recognized because they were too short or had unknown shape.

The pure clock-face design results in very low rate of unrecognized strokes and relatively high rate of wrong

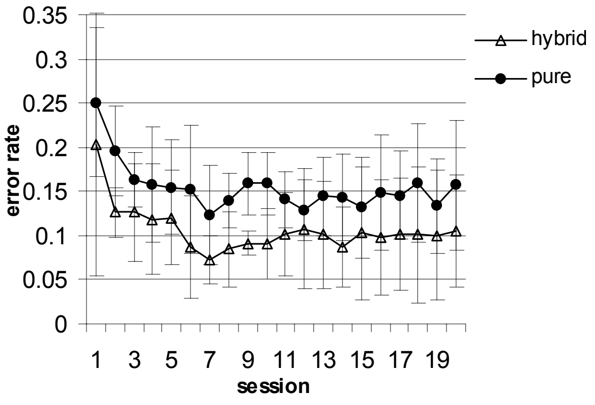


Figure 8. Error rate.

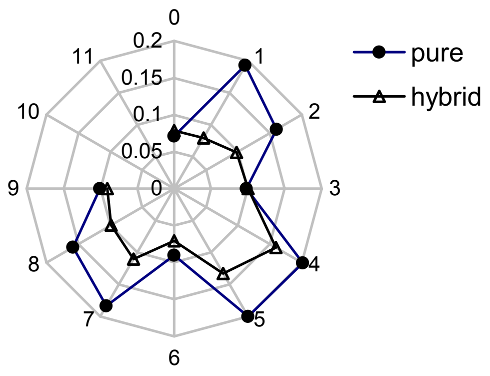


Figure 9. Average error rate per character.

input. The hybrid design on the other hand has mediocre error frequency in both error classes.

User preference

The end of session preference poll ended with total vote-count of 68 for the hybrid and 52 for the pure clock-face design. Five of the six users preferred the hybrid system after the first session. Four preferred the hybrid system after the last session. Given the small number of users, these results cannot be taken as anything more than a weak suggestion for possible general user preference for the hybrid system.

DISCUSSION

Error Rate

In comparison to the results of McQueen *et al.*, we measured a higher error rate for the pure clock-face design. The error rate for the hybrid design during the last three sessions was 9.8% which is close to 8.2% reported by McQueen *et al.* Overall, using a handheld finger operated touchpad instead of fixed stylus operated one seems to increase the error rate.

The task in our experiment was not exactly the same that McQueen *et al.* used. They allowed all input of 5 characters in length, whereas we allowed only correct input. This difference may have affected user behavior in a way that caused different error rates to be recorded. Although we were aware of this possibility, we felt that a cleaner

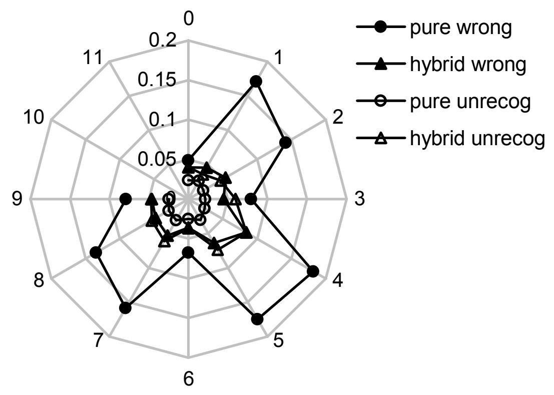


Figure 10. Error types for both systems.

definition for an error was desirable. In particular, in some cases it is next to impossible to determine the exact number of errors in the set-up used by McQueen *et al.* For instance, the user may have left out a character after which character per character based error detection is incorrect if the user follows the given string successfully after the missing character. The problem becomes even more challenging if the user notices that she is out of the sync and resynchronizes herself in the middle of the task. Our solution is not perfect. Real world applications rarely know what user is supposed to enter and therefore do not behave like our software. Our practice, however, reduces the ambiguity in error counting and simplifies the post-processing of the experiment logs.

One of the complaints that we got from the users was that the hybrid system is sometimes too picky in its recognition. Characters that the users thought were OK were not recognized. After the experiment we reviewed our character set and found that we could indeed relax some of the parameters given to the recognizer. We reran the data recorded in the hybrid condition through the recognizer. On average the number of unrecognized characters per session decreased by 25 of which 7 were moved to the wrong character category and 18 to the correct character category. This changed the overall error rate for the hybrid system from 10.2% to 9.3%. The percentage of unrecognized hybrid strokes sank to 3.6%. This is getting close to 2.6% measured for the pure clock-face design. This suggests that there are very few serious attempts at number entry left within the unrecognized hybrid category. Instead it consists mainly of accidental taps on the touchpad and truly malformed strokes.

Speed

Our results for overall entry time per character by the end of the experiment are about 100 ms longer than what McQueen *et al*. reported for the PiePad. This can be partly attributed to the higher error rate and the cost that errors had in our experiment. In the PiePad experiment users did not need to react if they made a substitution error (by far the most probable kind of error with PiePad). In our case the task would not proceed before correct input was received. This meant that the user had to notice that an error occurred and re-enter the character. This added cognitive involvement might have slowed our users down in comparison to the users of McQueen *et al*.

Another factor that may explain slower speed in our experiment is the pacing of the task. We required a continuous 15-minute effort whereas McQueen *et al*. had breaks after a block of ten 5-number groups. The rationale behind this change in the procedure was aimed to tease out differences between the systems caused by fatigue during the rather demanding 15-minute effort. However, we found nothing conclusive in the resulting data.

McQueen *et al*. excluded the first number in each group of 5 from their analyses because the preparation time for that number is confounded by the effort of perceiving the whole group that just appeared. We excluded only the first character in the session. This procedure may have the effect of increasing the preparation times for one fifth of our data. This is not a problem in the within study comparison that we were mainly interested in, but may explain some of the difference between our results and those reported by McQueen *et al*.

Handwriting time can be modeled with some accuracy based on the geometric form of the characters [2, 3, 10]. Applying the model given in [10] yields a complexity estimate of 1 for the straight lines and 2 for the curved ones. This means that the average writing time for the hybrid characters should be 0.6x2+0.4x1=1.6. This is 1.6 times the estimate of 1 for the pure method. This ratio is very different from the 1:1 ratio that we measured for the total writing time. The explanation is that the model only models the scripting time, not the preparation time and, therefore, not the total writing time. Our results do show a difference in scripting time (see Figure 7), but the difference, on average, is only about 30 milliseconds, which translates to ratio of 1.15 between the systems. A condensed summary on these computations is given in Table 1.

A closer inspection based on the Steering Law [2] reveals that the smaller than expected difference can be partly explained by the wider imaginary menu slices used in the hybrid method. Assuming that the number entry task can be modeled with the basic Steering Law goal passing tasks, the scripting time should follow linearly the Steering Law index of difficulty computed as $\log_2(A/W+1)$, where A is the length of a goal passing task and W is the width of the goal. We choose W to be the width of the slice at the end, because we can assume that at the beginning the finger is always perfectly positioned therefore always hitting the first target. The index of difficulty for the strokes in the pure condition is roughly $\log_2(2/1+1)=1.58$ bits. The difficulty of the straight strokes in the hybrid condition is the same 1.58 bits. The acceptable proportions of the beginning of the curved strokes, however, are different from the straight strokes. They may deviate 30 degrees to both directions

Table 1. Index of difficulty estimates for the systems.

	First Part	Second Part	Weighted Mean	Ratio
Geometric form of characters [10]				
Pure	1		**1**	1.6
Hybrid	1	1	**1.6**	
Steering law [2]				
Pure	1.58		**1.58**	1.37
Hybrid	1.58	1	**2.18**	

making the width of the slice 60 degrees, which yields index of difficulty of 1. A worst-case scenario for the latter part of the curved strokes is a 30-degree slice (same as all strokes in the pure condition. Thus, the overall index of difficulty for the curved strokes is 1+1.58=2.58 bits. The average index of difficulty for the hybrid characters is 0.6x2.58+0.4x1.58=2.18. This yields a scripting time ratio of 2.18/1.58=1.37 between the two systems. This is close to the 1.15 ratio that we observed. Because the index of difficulty for the curved strokes is based on a worst-case assumption, the observed ratio of scripting times should be smaller than what was computed above.

Even more detailed models of human hand motion would not help in describing the number entry speed very accurately. This is because hand motions are not the only factor affecting writing speed. In our experiment the other factors are represented by the preparation time. It accounts for about two thirds of the total character entry time. Unfortunately, we do not have models for predicting preparation time based on the shape of the characters.

CONCLUSIONS

Our experiment suggests that the hypothesis on the benefits of the lower error rate and better performance exhibited by the hybrid method may indeed be true. The error rate we measured for the pure clock-face design used with a handheld touchpad is higher than what McQueen *et al.* measured for stylus operated fixed tablet. Therefore strategies for reducing the error rate are more than welcome. The hybrid design seems to reduce the error rate significantly. However, the 9.8% error rate we measured during the last three sessions is still higher than most users would be willing to accept in general text entry situations [7]. All in all, we have demonstrated that the clock-face metaphor can be transferred to touchpad use without significant loss of entry speed or increase in error rate.

ACKNOWLEDGMENTS

We wish to thank the volunteer test subjects for their effort, Scott MacKenzie for inspiration, Roope Raisamo, and several other researchers at TAUCHI for valuable discussion. The work of the first author was financed by the Tampere Graduate School in Information Science and Engineering. This work was supported by the Academy of Finland (project 173987).

REFERENCES

1. 3Com Corporation, *Graffiti.* http://www.palm.com/products/input/
2. Accot, J., and Zhai, S. Beyond fitts' law: Models for Trajectory-based HCI Tasks. *Proc. of the CHI 97*, 295-302, ACM, 1997.
3. Accot, J., and Zhai, S. Performance Evaluation of Input Devices in Trajectory-based Tasks: An Application of the Steering Law. *Proc. of the CHI 99*, 466-472, ACM, 1999.
4. Bangay, S. Cirque Cat Driver for Linux. Available at http://cs.ru.ac.za/homes/cssb/cirque/
5. Callahan, J., Hopkins, D., Weiser, M., and Shneiderman, B. An Empirical Comparison of Pie vs. Linear Menus. *Proc. of the CHI 88*, 95-100, ACM, 1988.
6. e-acute, Octave User's Manual. Available at http://www.e-acute.fr/English/manual/manualV1.html
7. LaLomia, M.J. User acceptance of handwritten recognition accuracy. *CHI 94 Conference companion*, 107, ACM, 1994.
8. Enns, N.R.N., and MacKenzie, I.S. Touchpad-based Remote Control Devices. *CHI 98 Summary*, 229-230,ACM, 1998.
9. Goldberg, D., and Richardson, K. Touch-typing With a Stylus. *Proc. of the CHI 93*, 80-87. ACM, 1993.
10. Isokoski, P. Model for Unistroke Writing Time. *CHI Letters: Human Factors in Computing Systems, CHI 2001,* 3(1): 357-364, ACM, 2001.
11. Isokoski, P., and Raisamo, R. Device Independent Text Input: A Rationale and an Example. *Proc. of the Working conference on Advanced Visual Interfaces AVI2000*, 76-83, ACM Press, 2000.
12. Kurtenbach, G., and Buxton, W. User Learning and Performance with Marking Menus. *Proc. of the CHI94*, 258-264, ACM, 1994.
13. Kurtenbach, G., and Buxton, W. The Limits Of Expert Performance Using Hierarchic Marking Menus. *Proc. of INTERCHI '93*, 482-487, ACM, 1993.
14. Kurtenbach, G., Sellen, A., and Buxton, B. An Empirical Evaluation of Some Articulatory and Cognitive Aspects of Marking Menus. *Human-Computer Interaction*, 8, 1-23, 1993.
15. MacKenzie, I.S., Nonnecke, B., Riddersma, S., McQueen, C., and Meltz, M. Alphanumeric Entry on Pen-based Computers. *International Journal of Human-Computer Studies*, 41, 775-792, 1994.
16. Mankoff, J., and Abowd, G.D. Cirrin: a Word-level Unistroke Keyboard for Pen Input. *Proc. of the ACM UIST 98*, 213-214, ACM, 1998.
17. McQueen, C., MacKenzie, I.S., and Zhang, S.X. An Extended Study of Numeric Entry on Pen-Based Computers. *Proc. of Graphics Interface '95*, 215-222, Canadian Information Processing Society, 1995.
18. Perlin, K. Quikwriting: Continuous Stylus-based Text Entry. *Proc.of the ACM UIST 98*, 215-216, ACM, 1998.
19. Venolia, D., and Neiberg, F. T-Cube: A Fast, Self-disclosing Pen-based Alphabet. *Proc. of the CHI 94*, 265-270, ACM, 1994.