PressureText: Pressure Input for Mobile Phone Text Entry

David C. McCallum
University of Manitoba
Computer Science
Winnipeg, Canada
dmccallum9@gmail.com

Edward Mak
University of Manitoba
Computer Science
Winnipeg, Canada
ed.mak.is@gmail.com

Pourang Irani
University of Manitoba
Computer Science
Winnipeg, Canada
irani@cs.umanitoba.ca

Sriram Subramanian
University of Bristol
Computer Science
Bristol, UK
sriram@cs.bris.ac.uk

Abstract

Pressure sensitive buttons are appealing for reducing repetitive tasks such as text entry on mobile phone keypads, where multiple key presses are currently necessary to record an action. We present PressureText, a text-entry technique for a pressure augmented mobile phone. In a study comparing PressureText to MultiTap, we found that despite limited visual feedfback for pressure input, users overall performed equally well with PressureText as with MultiTap. Expertise was a determining factor for improved performance with PressureText. Expert users showed a 33.6% performance gain over novices. Additionally, expert users were 5% faster on average with PressureText than MultiTap, suggesting that pressure input is a valuable augmentation to mobile phone keypads.

Keywords

Pressure input, multi-level button, mobile phone text-entry.

ACM Classification Keywords

H.5.2 Input devices and strategies

Introduction

Recent studies have demonstrated the benefits of using pressure sensitive input. Researchers have integrated pressure sensors into existing devices [1,2,5,7], proposed new pressure-based techniques [3] and have explored methods for overcoming some of the limitations with pressure input [6]. What is particularly appealing about pressure interaction is the ability to easily select from a large number of input states [1,3,5,7].

One application area that can benefit from pressure interaction is mobile phone text entry. Most commercial mobile phones assign several alphanumeric characters to each key. For example, the two key also has the characters `d`, `e`, and `f` assigned to it. In the standard MultiTap technique, users press a button multiple times to select the desired character. An average phrase of seven words requires nearly 70 key presses with MultiTap [3]. However, applying pressure interactions to text entry is challenging. First, prior pressure based techniques require displaying full pressure values, a task that is difficult to do on all mobile displays [5,7]. Second, the most effective pressure selection techniques consume at least 1 sec for each selection [5,7], which is inappropriate for mobile text-entry. We addressed these challenges in this paper.

We present PressureText, a technique that uses pressure input for text entry on the standard 12-button mobile phone keypad. In PressureText, each button on the keypad is a sensor capable of producing a range of pressure values. PressureText maps pressure ranges to each of the characters on a key (Figure 1). A soft press invokes the first character, a medium press the second and a firm press the third character. A firm press with a short dwell invokes the fourth character on keys containing four characters.

We describe the design of PressureText and the results of an experiment that provide insight into some of the favorable PressureText design decisions. Our results suggest that accurate and rapid text-entry with pressure input is possible after repeated exposure.

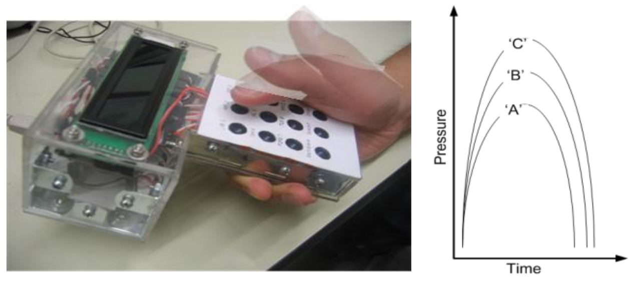


Figure 1. Left: A user applies varying amounts of pressure to select a letter. Right: the letter `c` requires more force than `a`.

Pressure Text

To harness the potential of pressure for text entry, we consider the discretization of the pressure space, pressure control, and character selection.

Discretization

The choice of discretization function plays a key role in reducing error rates [7]. We made use of the entire pressure space, which has led to a discretization function based on a logarithmic function along with a debouncing factor (explained below). In the logarithmic function, the lower boundary of each level is defined as $log_b(level\ number)$, where the level number ranges from 1 to 3 (at most 3 levels on each key) and b is the log base, which we vary between 2 and 3.15. Higher values of b allocate larger amounts of space to the lower pressure levels.

Debouncing

Pressure based interactions suffer from inadvertent jitters or crossings from the users' fingers or hand. To reduce these effects Shi et al. proposed the use of a fisheye function to "lock" the user onto a specific pressure level [7]. Prior studies have shown that the fisheye function can outperform other forms of discretization functions and reduce errors [7]. We implemented a debouncing algorithm with a similar effect. We set thresholds at the boundaries between adjacent levels, to prevent oscillation between them. To cross from one level to the next, the pressure values must not only cross the boundary, but also the threshold surrounding it. In our experiments the threshold was set to 10% of the level size, in pressure units.

In Figure 2 we depict the debouncing thresholds and the effect of applying pressure when moving from one level to the next. The dark lines are thresholds. The shaded area is the currently selected level, including its debouncing threshold. Debouncing operates similar to the fisheye in [7] but also tailors the radius of the fisheye proportionally to the amount of pressure allotted to a level.

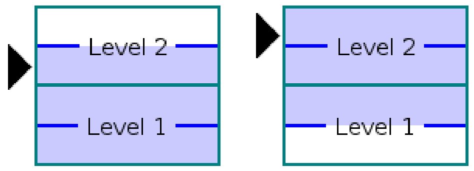


Figure 2. Debouncing visualization. Shaded area shows the currently selected level along with its threshold. The triangle represents the pressure level. Left: Pressure within Level 1's threshold. Right: pressure beyond Level 1's threshold, selecting Level 2.

Letter Selection

A user scans a character by applying sufficient pressure to reach the level for its associated character. Scanned characters are displayed on the LCD. Prior work on pressure input has relied on two design strategies for controlling and invoking the selection of a pressure level [5],[1]. The first design element consists of the use of full visual feedback. Here, the user is shown their current pressure value within the entire pressure range. However, complete visual feedback is impractical on mobile devices due to the small display size. Furthermore, text entry requires rapid movements that occur in a shorter time span than what is accessible with complete visual feedback. A second strategy for correctly invoking a selection involves the use of a delimiter such as a rapid release of the pressure sensor [5], a dwell on the appropriate pressure level [1],[5], or the use of an auxiliary button [1],[7]. In the context of text entry, an auxiliary button is impractical and dwelling is significantly time consuming. Based on prior results, we adapted quick release to operate in a context suited for text entry.

Our mechanism scans to the character associated with the inputted pressure value. When the button is released, the character is typed. If the user applies additional pressure to move to a higher level, we scan to the associated character immediately. When the user reduces the amount of pressure to move to a lower level, we scan to the new character after a 250 ms delay. This technique implicitly provides a partial implementation for bi-directional pressure input, a concern that is not entirely resolved [6]. Additionally, this technique prevents users from scanning to the lowest possible level whenever a button is released.

On a standard cell phone layout, there are two keys with four characters: PQRS and WXYZ. Interacting with four pressure levels is error-prone [7], and we restrict all buttons to only three pressure levels. To enter the fourth character, the user scans to the third character and dwells for 750 ms. For example, to enter s, the user scans to r and waits briefly. The scan and dwell are considered to be two separate gestures. Once the fourth character has been selected, the user cannot scan to other characters. The character is typed when the button is released.

Experiment

To evaluate our design choices for PressureText, we compared it to MultiTap. MultiTap is still commonly used on mobile phones (for various functions) and has often served as a baseline for comparison to new techniques in other studies [9].

Apparatus

We developed a pressure sensitive keypad (Figure 1). The buttons were laid out with a spacing of 1.5cm between columns and 1.75cm between rows. We used pressure sensors (#IESF-R-5L from CUI Inc.) that could measure a maximum pressure value of 1.5N and provided 1024 pressure levels. The device was connected to a desktop computer with USB.

Participants

Nine university students, three female and six male volunteered for the experiment. Two were left-handed. Participants received a small compensation. Five participants used MultiTap as their default text-entry mechanism on their cell phones and two others used T9. The remainder did not use a cell phone for texting very often. Of the nine participants, three were considered experts as they had used the pressure keypad more frequently than all the others. At most, the experts obtained an additional hour of training with the device through initial pilot studies.

Procedure

We employed the unconstrained text entry evaluation paradigm [3,8]. In this method, the participant enters a short phrase which appears on an auxiliary screen. The LCD on the device showed the text as users typed it in. The primary instruction was to proceed quickly and accurately. In our study, participants entered phrases from the corpus provided by Soukoreff and MacKenzie [8]. We started and ended a timer when the user entered the first and the last character, respectively. The phrase completion time incorporates the time to correct errors using the backspace key. Participants had several practice trials before they began the experiment. Participants were instructed to enter text with only the thumb of their dominant hand. Text entry included the use of the SPACE and BACKSPACE keys.

Design

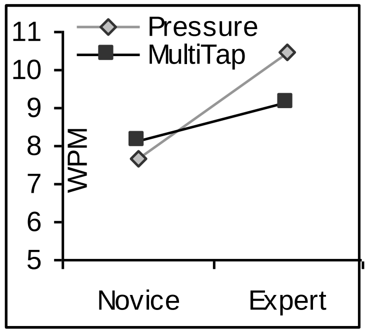
Prior studies have reported asymmetric transfer effects when comparing different text-entry mechanisms [9]. To prevent asymmetric transfer effects, we used a mix-design that was carried over three different sessions, each lasting an hour. The same participants performed all three sessions. Consecutive sessions for any participant were at least 24 hours apart. The first session used a within-subjects design. Each participant performed two blocks with both PressureText and MultiTap. In the second session half the participants (randomly selected) performed five blocks of MultiTap, and the other half performed five blocks of PressureText. In the third session, the participants who performed MultiTap on the previous day performed the experiment with PressureText, and vice versa.

Each block began with four practice phrases, followed by ten different phrases selected randomly from the corpus. Phrase selection for each of the two blocks were done before the experiment, and presented in the same order to each participant. All participants entered identical phrases in the same order, the only difference being which technique they used. In summary, the design was as follows: 2 techniques × 9 participants × 7 blocks × 10 phrases per block (excluding practice phrases) = 1260 phrases in total. The entire experiment lasted three hours over three days.

Results

Text Entry Speed

We use words-per-minute (WPM) as the primary measure to evaluate text entry speed. WPM is calculated as characters per second×60/5. PressureText had a higher average WPM (9.1) than MultiTap (8.64 WPM). Figure 3 shows average WPM for expertise level and block number.



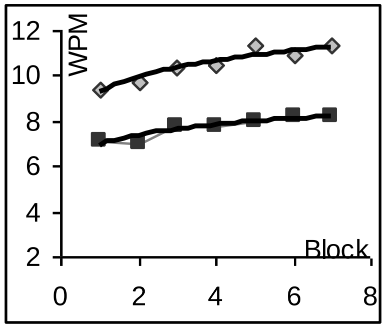


Figure 3. Left: WPM for PressureText and Mutlitap; Right: learning curve for PressureText and MultiTap for experts.

A univariate analysis of variance did not show a significant difference in text entry speed for *technique* ($F_{1,14}$=.39, p=.542) but revealed a significant effect for *expertise* ($F_{1,14}$=5.24, p=0.037) on text entry speed. The average text entry speed for experts was 9.83 WPM (sd=.101) and 7.8 WPM (sd=.075) for the novice group. We did not observe any significant *technique*×*expertise* interaction effect. We found that performance improved over the number of blocks with both techniques. The improvement over blocks was particularly noticeable for experts (Figure 3, right).

Error Rates

We computed the average corrected error rate (the user types an incorrect letter and then corrects it) as suggested in [3]. Overall the average error rate with PressureText was 8.6% compared to 2.7% with MultiTap. Interestingly, we noticed a larger decrease in error rates over multiple blocks with PressureText (16.3% for the first block, compared to 8.7% in the 7th block) than with MultiTap (5.4% in the 1st block, vs. 2.8% in the final block). We also observe that while error rates with MultiTap begins to reach a plateau between block 5 and block 7, error rates with PressureText continue to decrease even over the last few blocks. Characters that were placed in the middle range of the discretization (`b`, `k`, `x`) resulted in the highest number of errors.

Discussion

Our results suggest that PressureText is a suitable augmentation to existing mobile phone keypads. To facilitate text entry with pressure input, users need a certain level of expertise. To facilitate learning, pressure based keypads could be integrated with simple games as seen in other environments. Based on our results and performance in the experiment, we suspect participants will be able to learn text entry with pressure input fairly quickly.

As with other pressure based input technique, we also observed a slightly higher number of errors with pressure. We believe this factor was key in slowing users' typing speed and was primarily related to expertise. However, this is not surprising, considering that our setup is distinct from prior work in that we did not provided any visual feedback, as in prior work [1,4,5,7]. We only displayed the characters to be entered and provided no assistance to the user with respect to their current pressure values. Furthermore, our results suggest that the highest amount of error occurred in the middle letter of the pressure range, i.e. entering the letter `b` was more error prone than entering letter `a` or `c`. Improved designs would require a refinement on the discretization function, the debouncing thresholds or other potentially influential variables to reduce the amount of errors.

Conclusion

We introduce PressureText, a pressure-based input technique for the common task of entering text on mobile phone keypads. Based on these constraints we introduce various design elements, including a debouncing function to reduce errors. Results show that users have a marginally higher word per minute with PressureText than MultiTap. We observe that performance with pressure improves with expertise. Reducing errors with pressure input will facilitate even higher rates of text entry. In general pressure augmentation is a valuable addition to current mobile phone keypads. Pressure could be used in conjunction with techniques such as T9 (to disambiguate word selection) and for other tasks, such as browsing documents, on mobile phones. In future work we will continue to improve aspects of pressure text input, with multi-modal feedback such as vibrotactile cues.

References

[1] Cechanowicz, J., Irani, P., Subramanian, S., Augmenting the mouse with pressure sensitive input. Proc. CHI 2007, 1385-1394.

[2] Edward C. Clarkson, Shwetak N. Patel, Jeffrey S. Pierce, and Gregory D. Abowd 2006, Exploring Continuous Pressure Input for Mobile Phones, GVU Tech. Report; GIT-GVU-06-20. http://hdl.handle.net/1853/ 13138.

[3] MacKenzie, I. S., and Tanaka-Ishii, K. (Eds.) Text entry systems: Mobility, accessibility, universality, pp. 332. San Francisco.

[4] Ramos, G. and Balakrishnan, R., Zliding: fluid zooming and sliding for high precision parameter manipulation. *Proc. UIST 2005*, 143-152.

[5] Ramos, G., Boulos, M., and Balakrishnan, R., Pressure widgets. *Proc. CHI 2004*, 487-494.

[6] Rekimoto, J. and Schwesig, C., PreSenseII: bi-directional touch and pressure sensing interactions with tactile feedback. Proc. CHI 2006 Extended Abstracts, 1253-1258.

[7] Shi, K., Irani, P., Gustafson, S., Subramanian, S. (2008) PressureFish: a method to improve control of discrete pressure-based input. *Proc. CHI'08*, 1295-1298.

[8] Soukoreff, W., & MacKenzie, I.S. (2002). Text entry for mobile computing: Models and methods, theory and practice. *Human-Computer Interaction*, 17. p. 147-198.

[9] Wigdor, D. and Balakrishnan, R. (2003) TiltText: Using tilt for text input to mobile phones. *Proc. UIST '03*, 81-90.