The Unigesture Approach
One-Handed Text Entry for Small Devices

Vibha Sazawal[1], Roy Want[2], and Gaetano Borriello[1,3]

[1] University of Washington, Box 352350
Seattle, WA 98195-2350 USA
+1-206-543-1695
+1-206-543-2969 (fax)
vibha, gaetano@cs.washington.edu
[2] Intel Corporation, 2200 Mission College Blvd
Santa Clara, CA 95052-8119 USA
+1-408-765-9204
roy.want@intel.com
[3] Intel Research Seattle, 1100 NE 45th St
Seattle, WA 98105-4615 USA
+1-206-545-2530

Abstract. The rise of small modern handhelds mandates innovation in user input techniques. Researchers have turned to tilt-based interfaces as a possible alternative for the stylus and keyboard. We have implemented Unigesture, a tilt-to-write system that enables one-handed text entry. With Unigesture, the user can jot a note with one hand while leaving the other hand free to hold a phone or perform other tasks. In this paper, we describe a prototype implementation of Unigesture and outline results of a user study. Our results show that individual variations among users, such as hand position, have the greatest effect on tilt-to-write performance.

1 Introduction

Mark Weiser [1] defined a vision of ubiquitous computing that replaced the one-size-fits-all PC model with a variety of task-specific devices of different sizes. The "tab", a small credit-card sized device, the "pad", a tablet-sized device, and the "board," a whiteboard-sized device, would be peppered about a room, with tabs being the most plentiful.

The tab envisioned by Mark Weiser is now reality. The Hikari [2], designed at Xerox PARC, measures 80 × 64 × 14 mm. The Itsy [3], designed at Compaq, measures 118 × 64 × 16 mm. Both of these super-small computers have been exciting test beds for research, as well as useful devices in their own right.

As noted in [4] and [5], devices the size of the Hikari or smaller may be too small for traditional user input techniques. The Hikari has 4 buttons and smaller devices (so-called keychain computers [4]) are likely to have even less. There is no keyboard and little screen space on which to make use of a stylus. Entering text on these devices, despite their increasing computational power, is incredibly awkward.

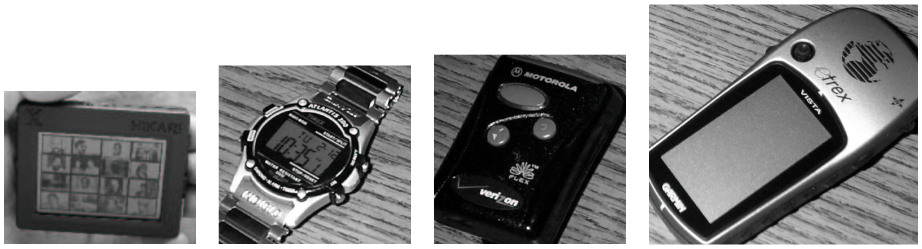


Fig. 1. Small devices with few or no buttons and varying amounts of screen space. User input on such devices can be very awkward. Pictured is the Hikari, a Timex Atlantis 100 TM watch (approx. 41mm in diameter), a Motorola TM BR850 pager (approx. 60 × 40 × 15 mm), and a Garmin eTrex Vista TM GPS receiver (112 × 51 × 30 mm).

Designers of the Hikari, the Itsy, and other small devices have turned to embodied user interfaces as a potential replacement for traditional techniques. Embodied user interfaces tightly couple the I/O of a device with the device itself. For example, a person may interact with a device by physically manipulating it – squeezing it, pushing it, and so forth. Devices such as the Hikari and the Itsy make use of an accelerometer that detects how much the device is tilted. Tilt-to-scroll and tilt-to-select have become useful input techniques for these devices.

In this paper, we tackle the problem of text entry for super-small devices. Text entry is a very important requirement for small device users. In December 2000, fifteen billion text messages were sent using phones alone [6]. As devices shrink in size, the challenges of enabling text entry grow. For example, users of the Garmin eTrex Vista pictured in Fig. 1 currently use the one button on the top left to enter text, which is incredibly cumbersome.

But what are designers to do with such small devices? There is no space for more than a couple buttons, and the small amount of screen space complicates use of a stylus. In addition, use of the stylus requires two hands, one to hold the small device and the other to hold the stylus. Two-handed use forces the user to discard whatever other objects the user was manipulating just to jot a note. A one-handed approach to text entry is an exciting alternative. With one-handed entry, the user's second hand is free to hold something else or perform other tasks.

Tilting has proved to be a successful mechanism to enable selection on tab-like devices, and so we decided to apply a similar approach to text entry. A tilt-based approach to writing offers an alternative to the stylus when screen space is small or nonexistent. Tilt-to-write also offers a one-handed method for text entry.

2 Unigesture: A Tilt-To-Write Method

Tilt-to-write is similar to tilt-to-select, but there are some important distinctions that stem from the differences between writing and selecting from a finite list. The first is a difference between the number of commands entered in a single sequence. Even a short phrase can contain 8+ letters, whereas one is unlikely to select from 8+ finite lists all in a row. The second is the speed at which writing is expected to occur. People expect writing to occur at the speed of typing, whereas selecting need only occur at the speed of a mouse-based scroll bar. Thus, we have a problem – users will enter far more commands with tilt-to-write than with tilt-to-select, but users will expect their commands to be processed much faster than ever required by tilt-to-select.

Thus, we concluded that the most obvious implementation of tilt-to-write, where one had a list of 26 letters, and one tilted to scroll through that list, would be unacceptable. Even with word completion, such a system is likely to be cumbersome. Instead, we needed to produce a different scheme, one that could potentially offer speed, and would not strain the arm or wrist even after repeated commands.

For guidance we turned to successful methods for fast text entry in traditional systems. For example, the Quikwriting [7] method of stylus text entry abandons alphabet-like letters in favor of a zone-based text entry method. The letters are divided into zones. With the stylus, the user's first motion selects a zone, and the second selects a letter within the zone. Text entry on a cell phone is also a zoned input system. Here, each numeric button can be treated like a zone. There are 3 or 4 letters associated with each zone. One technique for writing with a phone involves pressing the associated button 1-4 times to reach the desired letter. For example, pressing the number "2" once writes "a", pressing the number 2 twice writes "b", and pressing the number 2 three times writes "c". Another approach, developed independently by Tegic Communications [8] and Dunlop and Crossan [9] [10], builds upon this simple phone text entry method. This new approach, commonly known as "T9" TM , requires only one press of each numeric button. When "2" is pressed, the phone has no idea whether "a", "b", or "c" is wanted. The phone must guess. As the user enters more letters, the phone can use a dictionary to help it make good guesses as to which word or partial word would best fit that pattern of button presses. By using inference, T9 manages to save the user from unnecessary button presses.

Unigesture applies these time-saving ideas to tilt-to-write. With Unigesture, the alphabet is divided into 7 zones. Space, or ' ', is allocated a zone of its own. To write a letter, the user tilts in the direction of a zone and then returns to center. An example is shown in Fig. 2. No additional gesture is needed to specify which letter is wanted within the zone. Rather, the Unigesture system will accept a sequence of tilt gestures and then attempt to infer a word.

Advantages of the Unigesture method include the variety of slight tilt gestures that can be used. Each tilt gesture returns the user to a neutral center position, which allows the arm and wrist to avoid strain. Slight tilts are also a rather quick motion, which suggests that Unigesture users should be able to write quickly once

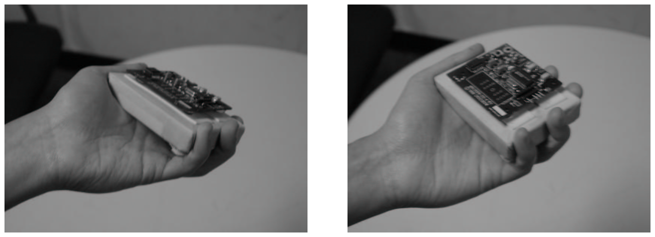


Fig. 2. Tilting a small device. Pictured is a user tilting up and then returning to center.

they have mastered the system. However, the inexperience we have with tilt-to-write systems results in a number of open design issues, each of which will have impact on Unigesture's possible success. Such design issues include the choice of a letter layout and the minimum amount of tilt acknowledged by the system. It is also not known how users will respond to a tilt-to-write system. We decided to build a prototype of Unigesture in order to explore these issues.

3 The Unigesture Prototype

The Unigesture prototype is implemented using an Analog Devices ADXL202EB evaluation board. The evaluation board is mounted to a block of plastic with similar physical characteristics to the Hikari. The board converts 2-dimensional accelerometer data into an RS232-compatible format, which is then read by software on a desk-top PC. The device mockup is shown in Fig. 3.

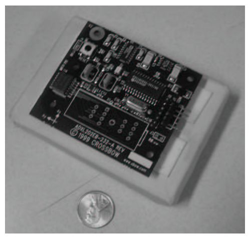


Fig. 3. Device mockup. The Analog Devices ADXL202EB evaluation board is mounted to a block.

The software on the PC recognizes the gestures and provides basic features for user feedback. The recognition algorithm is quite simple, intentionally so in order to be easily implemented on a small computer, no matter how primitive. The PC prototype is implemented as a multi-threaded Windows application in C++.

3.1 Recognition Algorithm

We considered two approaches to recognizing accelerometer data. The first is a "rolling marble" approach: the user is presented with a letter map and a marble located in the center of it. Accelerometer data is mapped to a point on the screen. The result feels like a "game" – to write a letter, one guides the marble into a visually displayed zone.

The "marble" approach has many advantages. This system is very easy to implement, it gives the user a lot of control, and it also can make the act of writing feel like a game. However, we decided not to apply this approach for two reasons. One, this approach requires a visual display as an integral part of the recognition algorithm. We wanted our approach to be implementable on a small tab that may have a very limited display. In addition, we were worried that the system would be too slow, because the system required the user to carefully guide the marble into a zone.

We decided to go with a recognition approach that was entirely independent of a visual display or screen coordinates. We approach the problem by examining the change in acceleration over a set of contiguous data points (a "window") recorded by the accelerometer. The system will attempt to identify a gesture as the window changes with time. The idea is that the user will develop haptic memory, associating each zone with a tilt of specific degree, duration, and direction. Because we consider changes in acceleration, the user can hold the device in any comfortable position and measurements are made relative to it.

The "window" approach is dictated by the values of 3 variables. The data is first smoothed, and thus the first variable is the length, l_{smooth}, of the smoothing window. Then the change in acceleration is measured, in this case by taking the first difference of the smoothed data. The second variable is the threshold, δ, by which the first difference data must exceed in order to remove noisy data. The third variable is the length, $l_{\text{recognize}}$, of the recognition window. The recognition window contains all of the first difference data that will be considered as a single unit. These three variables dictate the duration and degree of tilt that are required for recognition by the system.

For the prototype, we designed two versions of this algorithm. The first version, which we call the "deep tilt version", had a large l_{smooth}, a small δ, and a large $l_{\text{recognize}}$. The result was a system that required a long, deep tilt. The system never misrecognized accidental motions as deliberate tilts. However, the long, deep tilts were somewhat slow to perform. The second version, which we call the "slight tilt version", had a small l_{smooth}, a high δ, and a small $l_{\text{recognize}}$. The result was a system that required a short, fast slight tilt. The "slight-tilt" version required a good degree of coordination to operate; however, once learned, the user could write at a much faster speed.

Informal user studies caused us to refine our recognition algorithm. We found that users often made inadvertent motion directly after a deliberate motion. For example, a user would deliberately tilt up and then tilt back down to center, but then would tilt up and down inadvertently a few times before becoming stably still. To handle these unwanted oscillations, the system temporarily raises δ after every motion. The amount δ is increased is a function of the the degree of tilt formed when making the initial deliberate motion.

In the prototype, this algorithm is implemented using a pipe-and-filter architecture. Data from the accelerometer is first placed into a circular ring buffer. Each filter makes a transformation of the data and then places that transformed data into another circular buffer. The filters include a filter for smoothing the data, taking the first difference of the data, recognizing tilt motions from the data, and then recognizing zones from the tilt motions. A sequence of zones is then sent to the inference engine to produce a word.

3.2 Inference Engine

To reduce computational load, the system used a pre-computed dictionary tree that was keyed by zone sequence. In other words, the system would use a sequence of zones entered, which as a whole were intended to form a word, as a lookup key in the dictionary tree. The value returned for that key is an array of words that could all be formed given that same sequence of zones. The words and their ordering in each array were determined from sample texts. To reduce the search time, the array was ordered by the frequency with which each word occurred in the set of sample texts.

3.3 Letter Layout

For the prototype, we implemented two letter layouts. As stated above, the system had seven zones for characters. These zones can be named by direction: north, northwest, west, southwest, southeast, east, and northeast. The first letter layout placed commonly-used letters in zones that we believed were easier to reach. We had considered the cardinal directions: north, east, and west, to be the easier directions, with the diagonal directions somewhat harder. Thus, the first layout placed common letters in the north, east, and west directions, and uncommon letters in the diagonal directions. We refer to this layout as the “clustered layout”, as the common letters were clustered together.

The second layout was designed to reduce the number of possible words that could be formed from each zone sequence. In this second layout, each vowel is placed in its own zone. For each pair of zones, there is only one likely pair of letters that would appear consecutively in an English word. The second layout reduces the number of words associated with every zone sequence, but perhaps at the price of making some common letters more difficult to enter. We refer to the second layout as the “spread-out” layout as the common letters are spread out across the zones.

The two letter layouts are shown in Fig. 4.

3.4 User Interface and Feedback

The user interface provides a small amount of visual feedback. This feedback is intentionally limited so that it would be transferable to a small screen. Users are

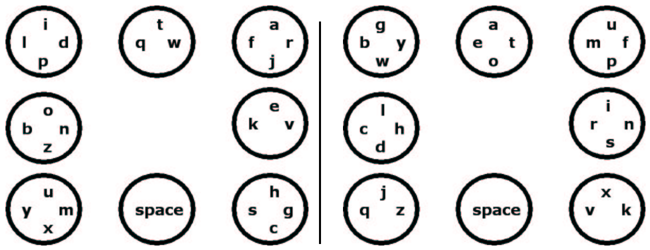
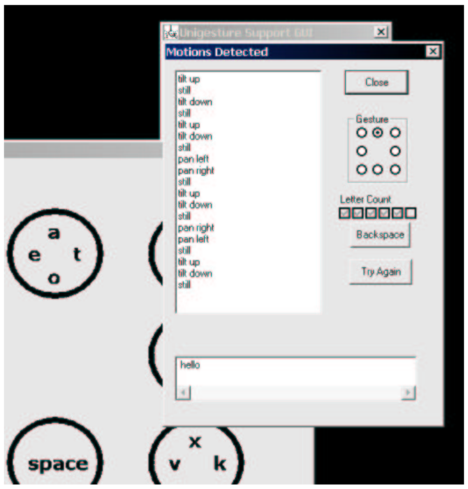


Fig. 4. Letter Layouts. Each layout consists of 7 zones for letters and the space zone. On the left side is the "spread-out" layout and on the right side is the "clustered" layout.

presented with a visual letter map. This map is static and separate from feedback, because the map is intended to be dropped as soon as the user memorizes it. Feedback from the system is implemented via a set of 8 radio buttons laid out in the same manner as the letter map and a set of check boxes that count the letters entered so far. A photo of the UI is shown in Fig. 5.



```
tilt up
still
tilt down
still
tilt up
tilt down
still
pan left
pan right
still
tilt up
tilt down
still
pan right
pan left
still
tilt up
tilt down
still
```

hello

Fig. 5. UI for Unigesture prototype. The user is writing "hello there". In this example, the "clustered" layout was used. The system was photographed just after the last 'e' in "there" was written. The five checkmarks and darkened north zone can be seen. The word "there" will appear on the screen after the user tilts down to select the space zone.

Users hold the mock handheld computer and tilt it. When a zone is recognized, the radio button associated with that zone darkens. A check box is also marked to show that a zone has been entered. As the user enters zones, the radio button of the current zone is darkened, while the number of checked boxes shows the number of zones that have been correctly entered so far.

After the user enters the zones for an entire word, the user inputs the “space” zone as a break. The zone sequence is then sent to the inference engine. The check boxes clear, and the user sees the word chosen by the inference engine. If the inference engine returns the word desired by the user, the user continues writing. Otherwise, the user clicks on the “Try Again” button. Clicking on the “Try Again” button repeatedly causes the system to traverse the array of words that match that zone sequence. One can easily imagine a more sophisticated system which remembered which words you write most often and would return them first. Other systems might allow you to manually set the frequency of words expected to be entered. For the purposes of our prototype, we did not need to implement these features, but one would want to implement them in a full-featured system.

The final piece of the UI worth noting is the “Backspace” button. Clicking on “Backspace” removes the last zone entered into the system. The last checkmark is removed as well, showing the user that the “Backspace” correctly took place.

On the PC prototype, the buttons are implemented as part of a Windows GUI and are selected via the mouse. In a full implementation, one would want these features to be buttons on the small device itself or implemented in some other fashion. For example, perhaps selecting the “space” zone multiple times could be used to tell the inference engine to “Try Again”.

It is important to note that even though there are differences between our prototype and a full implementation, the prototype still forms an excellent base on which to explore the design space for tilt-to-write interfaces. For example, if all the users in our study hold the mockup a certain way, then we have excellent information with which to guide button placement on the real device. On the other hand, if users all hold the mockup differently, then it may be better to invest in non-button interfaces to “Backspace” and “Try Again”. Because the prototype is clearly a mockup, users feel free to make suggestions, criticism, and comments, and feel freer with their use. This valuable interplay between user and device is often lost when users are given a “real device”, which feels official and unquestionable [11].

4 User Study

4.1 Experiment Setup

Once the prototype was complete, we conducted a user study to learn more about how a tilt-to-write system would perform in practice. The first user test consisted of 12 users, all employees from Intel Corporation in Oregon. All 12 users used a desktop computer for 4 or more hours a day and owned a cell phone that they used regularly. 7 of the 12 users also owned a handheld computer that they used regularly. We were interested in participants who were already familiar with computers, and in particular, with technology “held in the hand”.

The intent of the user study was to explore the design space of a possible tilt-to-write system. As described earlier, the Unigesture prototype has 2 letter map layouts ("clustered" and "spread-out") and 2 recognition versions ("slight-tilt" and "deep-tilt"). In our counterbalanced study, half the users received the "clustered" letter layout, while the other half received the "spread out" letter layout. Users were first trained in the Unigesture method. After their training, users were asked to write.

Because this method is intended for small devices, it was important to ask users to write messages of a realistic length. While users may appreciate the ability to jot down a quick note or the name of a GPS waypoint, they are unlikely to write their dissertation on a keychain computer. Thus, we defined 8 phrases, each containing between 10 and 12 characters. Users were asked to write these 8 phrases one at a time. Users were seated while they used the device.

Table 1. Phrases used in usability study

"on the web"	"thank you"	"be careful"	"try later"
"dont know"	"meet here"	"email jeff"	"how are you"

Half of these phrases were performed using the "deep tilt" recognition version and the other half with the "slight tilt" recognition version. Users would write the first 4 phrases using one version and then would write the latter 4 with the other version. They were permitted to write practice words to familiarize themselves with the new recognition algorithm if they chose.

Training consisted of the test-giver demonstrating how to use Unigesture, followed by the user trying out the device for a few practice words. Training was expedited if the user was already familiar with an inference system such as "T9". The user was allowed to practice as long as they wished, as long as the total training time was less than 20 minutes. The user was trained with either the "deep tilt" or the "slight tilt" recognition version, depending upon which version would be used for the first set of 4 phrases.

We tracked the user's mistakes and the time to complete each phrase. A mistake could have been one of two occurrences. The first is when the user tilts the device in a way that the system cannot recognize. When this happens, the user simply tries again. The second is when the system misrecognizes the user's tilts. When this happens, the user must press "Backspace" and then try again. Throughout the experiment, users were permitted to restart words if they chose. If the user restarted a word, only errors made during the last complete attempt of the word were counted.

As stated above, the purpose of the study was to explore the design space of a tilt-to-write system. We were interested in a number of questions:

1. Could all users write using the Unigesture method?

2. Would the performance of users vary with the level of tilt required by the recognition algorithm?
3. Would the performance of users vary with the choice of letter layout?
4. How would users react qualitatively to the Unigesture method?

The answers to these questions are explored in the next two subsections.

4.2 Quantitative Results

All 12 users could write phrases using the "deep-tilt" version of Unigesture. 10 of the 12 users could effectively write phrases using the "slight-tilt" version of Unigesture. Of these 10 users, 7 preferred the "slight-tilt" version, while 3 preferred the "deep-tilt" version. The 7 who preferred the "slight-tilt" version did so due to its speed and its ease on the wrist. The 3 who preferred the "deep-tilt" version preferred its accuracy.

Phrase-by-phrase, we can compare the number of errors made by users using the 2 layouts and the 2 tilt sizes. For the majority of phrases, there is not enough information to make statistically significant conclusions about the different versions. However, for one phrase, "try later", there was a significantly greater number of errors made using the "slight-tilt" version than the deep-tilt version. The table below shows the number of errors made by users when entering this phrase. Each number in the table refers to the quantity of errors made by a single user on the phrase; there were 3 users in each experimental condition. Statistical significance was determined by using a 2-factor analysis of variance (ANOVA) experiment [12] with the F-test at α=0.05.

Table 2. Number of errors made when writing phrase "try again"

	Slight-Tilt	Deep-Tilt
Clustered	0, 7, 8	1, 3, 0
Spread Out	4, 13, 4	1, 1, 2

For all phrases, the large bulk of variation in number of errors made per phrase occurs "within" treatments. In other words, variation due to the differences between each user had usually a greater effect on the number of errors made than the experimental setup that the user has been provided with.

We had initially believed that the diagonal zones would be more troublesome than the cardinal zones. However, this was not universally the case. Defining "trouble zones" as those zones that contained letters on which a user made 3 or more consecutive errors per phrase, 3 of the 12 users had problems with all 4 diagonal zones. But most users had trouble with a specific, customized set of zones, all based upon the way that the user held the device. For example, some right-handed users had trouble tilting to the right, and some left-handed users had trouble tilting to the left. 4 of the 12 users had troubles with the “space” zone.

4.3 Qualitative Results

Response to the Unigesture system was mostly positive. Users remarked that the system was “cool” and “clever”. Only one user had something negative to say; this user found the system “annoying”.

Users were asked which zones they found most troublesome. For the most part, the zones users identified as difficult were the ones where they had made their errors. However, users also reported that some zones felt awkward to select, but they were still able to enter a letter correctly.

Users with the “clustered” letter layout often had to click on the “Try Again” button in order to receive the word they had wanted. However, when asked, most users did not find the “Try Again” button annoying. The majority of users with the “clustered” layout felt that such a button was necessary, because they were giving the system incomplete information.

When finished with the experiment, we asked the user if they felt any physical discomfort. Of the 12 users, 2 users felt fatigued, and 1 user felt pain. These responses are definitely a concern. It is possible that users are exaggerating the tilt far beyond what they need to do to register a response with the system. Perhaps audio feedback or some other mechanism could be used to tell users to stop tilting. It is also possible that the device may need to be shaped in a more ergonomic form. Our study is too small to make any conclusive results, but it is clear that this concern is a primary issue for future work.

4.4 Numeric Map Experiment

We expected users in the experiment described above who performed well to still perform relatively slowly compared to an expert user, because the participants of the study were novices and had not memorized a letter map. In an attempt to measure the time spent looking at the letter map, we designed a second, smaller user study. In this study, we introduced a numeric layout that is not unlike a numeric keypad on a phone. (Numbers 5 and 9 are missing.) This numeric layout is quite easy to remember.

The experiment was designed as follows. The user was trained on a “slight-tilt” version of Unigesture and was given either the clustered layout or the spread-out layout. After training, users would write the first 6 phrases described in the earlier experiment, and they would also switch to the numeric layout and write two numbers. The experiment was counterbalanced.

Criteria for participation in this study was the same as the previous experiment. Of the 4 users who took part, 3 are employees at Intel Corporation in Hillsboro, Oregon and 1 is a computer science graduate student at the University of Washington in Seattle. The hypothesis was that the numeric layout would be familiar, and that users would perform faster when writing numbers.

The numbers to be written were also intended to be familiar for the majority of the participants – the first number, 264, is a common phone exchange for Intel in Hillsboro, and the second, 97124, is the zip code for Hillsboro.

As in the earlier experiment, the letter layout used, either clustered or spread-out, did not affect performance very much. Users achieved times as low as 9 seconds for a 3-letter number (plus the space character), which implies a maximum data speed for novices as high as 2-3 seconds per character. However, fast users also obtained times as low as 33 seconds for a 10 character phrase, which can be considered equivalently fast as the speed achieved when writing numbers. In other words, taking away the letter map and using something simpler did not appear to speed up users. This suggests that the amount of time spent searching for letters on the letter map is not as high as we had previously expected. The speed per character only marginally increased with the simpler numeric map, if at all.

Table 3. Average time to completion per character (seconds)

	User 1	User 2	User 3	User 4
Phrases	3.6	8.9	5.8	5.4
Numbers	3.5	8.2	5.3	10.4

If looking for letters in a letter map is not the problem, what is taking up a user's time? Review of the user study suggests that users had zones that particularly troubled them, and selecting letters from these zones took up time. In addition, some users make mistakes and failed to notice this. The user would then end up repeating an entire attempt to enter the number or phrase.

4.5 Discussion of the User Studies

To summarize, we found the following results:

1. Users can tilt-to-write, and most of them have the dexterity to handle the "slight-tilt" version. However, a major concern involves whether the method results in unacceptable strain on the wrist. Future work is needed to resolve this concern.
2. There was not a significant difference between the performance of users given the "clustered" letter layout and the "spread out" letter layout. Rather, users each had different sets of trouble zones and easy zones, and while these sometimes coincided with the zone layouts in the "clustered" letter layout or the "spread out" letter layout, this was usually not the case.
3. Users did not spend much time looking at the letter layout. Time is spent instead trying to master tilting in trouble zones.

The usability tests show that everyone holds handheld devices in a slightly different way. Thus, the ideal system is one that can be customized to operate only in the non-trouble zones for the user. It is very easy to imagine a system that identifies which zones are troublesome and then arranges letters so that these zones are avoided. An automated trainer could also easily determine whether the user can write with the slight-tilt version or should be switched to the deep-tilt version. This customization approach could have its drawbacks, though – if a user borrows another's handheld computer, they may be unable to write with it!

The differences among how people hold handheld computers also complicates button placement. We propose that non-button techniques for entering "Try Again" and "Backspace" be used. Possible options include non-tilt motions, such as shaking or squeezing, which will be easily distinguishable from the tilt motions used for zone entry.

Potential augmentation options also include audio feedback. After the nth letter, the device could state "Zone [direction] selected. This is the nth letter in the word." Audio tones could also be used. Another potential option is force or tactile feedback, which could increase the haptic memorization of tilt motions.

In addition, it is possible that common phrases could be mapped to tilt motions instead of letters. These phrases would allow users to write faster at the cost of expressibility.

4.6 Issues Not Addressed in the User Studies

There are three issues that were not explored in the user studies. The first is the dictionary maintained by the system. All the words that users were asked to enter were in the dictionary. In practice, there would have to be a mechanism to enter words into the dictionary, and there would also have to be a second means for entry to "fall back to" in case one is writing a URL or other non-word. The obvious fall-back mechanism to Unigesture would be a Bigesture system. In a Bigesture system, the user would enter tilt gestures in pairs. The first tilt motion would select a zone, and the second would select a letter in the zone. However, we do not know if the obvious choice is the right one.

Another issue involves training. The user studies involved a human trainer. However, we believe that training would be easy to automate. In practice, people grasped the concept of Unigesture quite quickly. When they made mistakes, they tended to make similar types of mistakes, although the particular zones involved may be different.

A third issue is the need to support the entry of letters, numbers and punctuation by an integrated system. The obvious choice is to have a number of character layouts, not unlike the Quikwriting system [7]. A button or shake could potentially toggle the user between the layouts.

5 Conclusion

Tilt-to-write is ideally intended for "tabs", small credit-card sized computers with few buttons and small or no screen space for a stylus. Experiences with a tab-sized mockup show that users can learn a tilt-to-write system and can effectively enter text using only the hand that is holding the device.

In this paper, we explored a number of potential parameters of a tilt-to-write system. We offered users different letter layouts and provided systems that required varying degrees of tilt. We found that the natural variation among users had a greater effect on the number of mistakes made than any tweaking of a system parameter. Thus, an ideal system would customize itself to the user.

There are a great deal of open questions in this exciting area. What is the best way to customize a tilt-to-write system to the user? How can tilting be mixed with buttons or other input techniques to provide effective hybrid forms of text entry? How useful is letter-by-letter entry versus entry of entire phrases via one tilt? Can we teach users to tilt-to-write without causing too much wrist strain? These questions guide our future work.

6 Related Work

There are a number of stylus-based methods for text entry on handheld computers, such as Unistrokes [13] and Quikwriting [7]. All of these techniques require that one hand holds the device and the other hand holds the stylus. They also require ample screen space. Writing methods that could be implemented via tilt-based gestures include Dasher [14] and MDITIM [15]. Both of these methods seem quite promising. However, in their current forms, Dasher requires a great deal of visual attention and MDITIM requires a large number of tilts per character. It is possible that modifications to these systems could make them excellent tilt-to-write interfaces. Nonetheless, we propose that the results we found for Unigesture will also hold for any other non-trivial tilt-based method – the system must be customized to avoid troublesome tilts in order to be successful.

Although tilt-to-write is a new concept, the idea of tilt-based user interfaces is by no means new. Original work in this area was done by the University of Toronto's Chameleon project [16], Jun Rekimoto's tiltable screen project [17], the Extreme User Interface projects of Xerox PARC [2], and the Shakepad project from MIT [4]. More recent work in the area of tilt-based cursors was done by Weberg et al. [18], who coined the analogy "A Piece of Butter on the PDA Display".

New techniques for speeding up text entry on handheld devices is also an active research area. Work exists in word [19] [20] [10] and "partial word" prediction [6]. These techniques complement the Unigesture approach.

7 Acknowledgements

We thank Kurt Partridge, Jeff Hightower, and Steve Swanson for their helpful advice. We also thank all participants of our user study for their time and thoughtful comments. This research was supported by a DARPA Expeditions grant N66001-99-2-8924. This work was performed in part at Xerox PARC.

References

[1] Mark Weiser. The computer for the 21st century. *Scientific American*, Sept. 1991.

[2] Kenneth Fishkin, Anuj Gujar, Beverly Harrison, and Roy Want. Embodied user interfaces for really direct manipulation. *Communications of the ACM*, Sept. 2000.

[3] William Hamburgen, Deborah Wallach, Marc Viredaz, Lawrence Brakmo, Carl Waldspurger, Joel Bartlett, Timothy Mann, and Keith Frakas. Itsy: Stretching the bounds of mobile computing. *IEEE Computer*, April 2001.

[4] Golan Levin and Paul Yarin. Bringing sketching tools to keychain computers with an acceleration-based interface. In *Proceedings of ACM CHI '99 Extended Abstracts*, 1999.

[5] Brad Myers, Scott Hudson, and Randy Pausch. Past, present, and future of user interface software tools. *ACM Transactions on Computer-Human Interaction*, March 2000.

[6] I. Scott MacKenzie, Hedy Kober, Derek Smith, Terry Jones, and Eugene Skepner. Letterwise: Prefix-based disambiguation for mobile text input. In *Proceedings of ACM UIST '01*, 2001.

[7] Ken Perlin. Quikwriting: Continuous stylus-based text entry. In *Proceedings of ACM UIST '98*, 1998.

[8] Tegic Communications. T9 web page. URL: http://www.t9.com.

[9] M.D. Dunlop and A. Crossan. Dictionary based text entry method for mobile phones. In *Proceedings of Mobile HCI 1999*, 1999.

[10] M.D. Dunlop and A. Crossan. Predictive text entry methods for mobile phones. *Personal Technologies*, 4(2-3), 2000.

[11] Yin Yin Wong. Rough and ready prototypes:lessons from graphic design. In *Proceedings of ACM CHI '92 Short Talks*, 1992.

[12] Richard Hogg and Johannes Ledolter. *Applied Statistics for Engineers and Physical Scientists*. Macmillan Publishing Company, 1987.

[13] David Goldberg and C Richardson. Touch-typing with a stylus. In *Proceedings of ACM INTERCHI '93*, 1993.

[14] David Ward, Alan Blackwell, and David MacKay. Dasher – a data entry interface using continuous gestures and language models. In *Proceedings of ACM UIST '00*, 2000.

[15] Poika Isokoski. A minimal device-independent text input method. Technical report, University of Tampere, 1999.

[16] George Fitzmaurice, Shumin Zhai, and Mark Chignell. Virtual reality for palmtop computers. *ACM Transactions on Information Systems*, 11(3), July 1993.

[17] Jun Rekimoto. Tilting operations for small screen interfaces. In *Proceedings of UIST '96*, pages 167–168, 1996.

[18] Lars Weberg, Torbjörn Brange, and Åsa Wedelbo-Hannson. A piece of butter on the pda display. In *Proceedings of ACM CHI '01 Extended Abstracts*, 2001.

[19] Fredrik Kronlid and Victoria Nilsson. Treepredict: Improving text entry on pda's. In *Proceedings of ACM CHI '01 Extended Abstracts*, 2001.

[20] T Masui. Pobox: An efficient text input method for handheld and ubiquitous computers. In *Lecture Notes in Computer Science (1707), Handheld and Ubiquitous Computing, Springer Verlag*, 1999.