From Letters to Words: Efficient Stroke-based Word Completion for Trackball Text Entry

Jacob O. Wobbrock[1,2] and Brad A. Myers[2]

[1]The Information School
University of Washington
Mary Gates Hall, Box 352840
Seattle, Washington 98195-2840
wobbrock@ischool.washington.edu

[2]Human-Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213 USA
bam@cs.cmu.edu

ABSTRACT

We present a major extension to our previous work on *Trackball EdgeWrite*—a unistroke text entry method for trackballs—by taking it from a character-level technique to a word-level one. Our design is called *stroke-based word completion*, and it enables efficient word selection as part of the stroke-making process. Unlike most word completion designs, which require users to select words from a list, our technique allows users to select words by performing a fluid *crossing gesture*. Our theoretical model shows this word-level design to be 45.0% faster than our prior model for character-only strokes. A study with a subject with spinal cord injury comparing Trackball EdgeWrite to the on-screen keyboard *WiViK*, both using word prediction and completion, shows that Trackball EdgeWrite is competitive with WiViK in speed (12.09 *vs.* 11.82 WPM) and accuracy (3.95% *vs.* 2.21% total errors), but less visually tedious and ultimately preferred. The results also show that word-level Trackball EdgeWrite is 46.5% faster and 36.7% more accurate than our subject's prior peak performance with character-level Trackball EdgeWrite, and 75.2% faster and 40.2% more accurate than his prior peak performance with his preferred on-screen keyboard. An additional evaluation of the same subject over a two-month field deployment shows a 43.9% reduction in unistrokes due to stroke-based word completion in Trackball EdgeWrite.

Categories and Subject Descriptors

H.5.2 [**Information interfaces and presentation**]: User interfaces — *Input devices and strategies*. K.4.2 [**Computers and society**]: Social issues — *assistive technologies for persons with disabilities*.

General Terms

Design, Experimentation, Human Factors, Theory.

Keywords

Word prediction and completion, word-level text entry, text input, goal crossing, unistrokes, gestures, trackballs, Fitts' law, Hick-Hyman law, Steering law, Zipf's law, EdgeWrite, WiViK.

ASSETS'06, October 22–25, 2006, Portland, Oregon, USA.

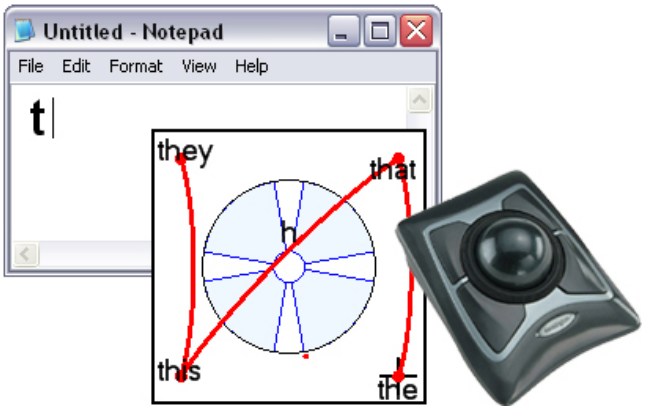

Figure 1. Trackball EdgeWrite with stroke-based word completion. A "t" has been made, and the current stroke is an "h", so "th-" completions are currently offered.

1. INTRODUCTION

Although many technologies exist for alternative computer access [7], studies show that less than 60% of people who need access solutions actually use them [9]. Furthermore, at least 35% of purchased solutions are never adopted [8]. Voice recognition systems, in particular, are subject to high abandonment rates [15]. Reasons cited for these failures include the high cost of devices, device complexity, and the need for extensive customization. Of prime importance, then, is *simplicity* in both the design of devices and in the process of adoption [8].

In an effort to provide a simpler desktop access technology, we previously introduced a method of desktop text entry for use with trackballs called *Trackball EdgeWrite* [30]. Unlike most trackball text entry methods, which require users to mouse over an on-screen keyboard, Trackball EdgeWrite uses *unistroke gestures*, which allow users to write "by feel" rather than "by sight." The result is a faster and less tedious method of trackball text entry for people who already use trackballs but cannot touch-type on a physical keyboard. Such users may have repetitive stress injuries, spinal cord injuries, arthritis, or some neuromuscular disorders.

Despite Trackball EdgeWrite's initial success compared to on-screen keyboards, it has been, until now, constrained to character-level entry. This is a limitation in many text entry systems, one which limits speeds. To address this problem, we introduce a major extension to Trackball EdgeWrite that takes it from a character-level method to a word-level one. Our extension is called *stroke-based word completion* (Figure 1), and it allows users to complete words using strokes instead of selecting words

from a list. Stroke-based word completion is a generalization of a more specific technique of ours intended for styli [29] which relied extensively on "pigtail loops" as in-stroke delimiters. Trackball EdgeWrite, on the other hand, uses no loops, relying only on straight line segments, and includes context-based word predictions along with fixed frequency-based word completions.

To our knowledge, this is the first word prediction and completion system that is stroke-based instead of pointing-based, and therefore designed to leverage *feel* rather than *sight*. This gives Trackball EdgeWrite significant advantages over on-screen keyboards. Our results for a subject with spinal cord injury show that stroke-based word completion provides a 46.5% increase in speed (8.25 *vs.* 12.09 WPM), a 36.7% decrease in errors (6.24% *vs.* 3.95% total errors), and a 43.9% reduction in strokes compared to his prior peak performance with character-level Trackball EdgeWrite. Furthermore, word-level Trackball EdgeWrite is 75.2% faster (6.90 *vs.* 12.09 WPM) and 40.2% more accurate (6.60% *vs.* 3.95% total errors) than our subject's prior peak performance with his own preferred on-screen keyboard, which he has used for 15 years. Now he uses Trackball EdgeWrite with stroke-based word completion instead.

2. RELATED WORK

2.1 Trackball Mousing

Most studies show that for able-bodied users, trackballs are slower and less accurate than conventional mice for pointing, dragging, and steering [2,18,19]. However, relative to other devices, trackballs perform reasonably well for short straight ballistic movements when *crossing a goal*. Capitalizing on goal crossing was the rationale behind the Trackball EdgeWrite design.

Despite their inferior performance compared to mice for able-bodied users, trackballs are preferred by many people [10,33]. Reasons include that trackballs do not require the wrist or forearm to elevate. They also do not require much space, making them suitable for placement in a user's lap or on a wheelchair tray. Rolling a trackball requires little strength, and if clutching is necessary, one must only lift one's finger or hand, not the device itself. Furthermore, trackballs are simple, readily available, robust, and cheap. A benefit of having an integrated text entry method for trackballs is that users who already use trackballs for mousing do not have to switch to other devices when entering text.

2.2 Trackball Text Entry

Prior trackball text entry methods have mostly used on-screen keyboards. Although on-screen keyboards are easy to learn, they have many drawbacks. For instance, they exacerbate mouse travel to and from a document. They introduce a second focus-of-attention such that a user's eyes cannot remain on his or her document [4]. They also require repeated target acquisitions for which trackballs are not well suited. Furthermore, they are visually fatiguing, equivalent to typing in a "hunt-and-peck" fashion. Finally, they consume screen real estate, considerably reducing one's visual workspace and increasing the need for window management. Note that although word prediction systems may increase speed, they do not alleviate these concerns.

In contrast to on-screen keyboards, a few relevant *stroke-based* text entry methods have been devised. Besides Trackball EdgeWrite [30], two other methods can be used with trackballs. One is *Dasher* [28], which allows any pointing device to enter text by moving through expanding letter regions whose sizes correspond to a letter's likelihood of entry. However, Dasher can be overwhelming because its letter regions continually rush toward the user. Another method is *MDITIM* [14], which defines letters using only the four cardinal directions. A drawback is that MDITIM's strokes generally do not resemble Roman letters.

2.3 Word-level Text Entry Methods

Researchers have noted that character-level entry is inherently limited [34]. As a result, recent attention has shifted to word-level techniques, in which single strokes or operations produce entire words. Cirrin [23] and Quikwriting [24] are two such techniques. In both designs, a person moves a stylus through fixed letter regions arranged around the periphery of a circle or square. These techniques are word-level in the sense that whole words are made in single (rather long) strokes, but each character within the word must still be acquired by the stylus.

An innovative approach to word-level stroking is *SHARK* [34], which presents a stylus keyboard over which strokes can be made. The shapes of these strokes are determined by the arrangement of letters on the keyboard. Users can gradually ramp from tapping words to stroking them, enabling higher speeds. This emphasis on *gradual* learning has been preserved in Trackball EdgeWrite's stroke-based word completion, since users can still stroke individual characters as they always have.

2.4 Word Prediction and Completion

A common approach to enhancing text input rates is to use word prediction and completion to populate a list with candidate words. Users select from the list to enter entire words or suffixes. Although the number of user actions is reduced, numerous studies show that additional perceptual and cognitive processes often make such systems *slower* instead of faster [11,16,25]. These findings highlight the challenge of designing effective word prediction and completions systems.

In the case of trackball text entry with an on-screen keyboard, candidate words appear as additional mouse targets, which further exacerbate mouse travel and the need for accurate target pointing. Although Anson *et al.* (2005) reported that word prediction and completion improved entry rates with on-screen keyboards, subjects reported high frustration because they disliked looking from their document to the word list and "felt that searching through the word list was tedious and distracting" [4]. With stroke-based word completion in Trackball EdgeWrite, we overcome the drawbacks of visually-intensive word selection by providing a gestural alternative that performs as well or better.

3. STROKE-BASED WORD COMPLETION

3.1 Brief Overview of Trackball EdgeWrite

Until now, Trackball EdgeWrite has provided a *character-level* means of writing with a trackball. Trackball EdgeWrite allows users to "pulse" the trackball toward the four corners of a virtual EdgeWrite square (Figure 2, next page). As users connect these corners in patterns similar to Roman characters, letters are produced [32]. Segmentation between letters is achieved when force (i.e. motion) ceases on the trackball.

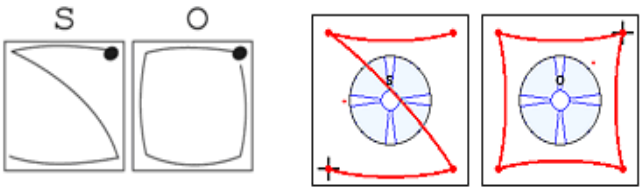

Figure 2. EdgeWrite gestures for "s" and "o". The stroke segments are arcs determined by the corners entered.

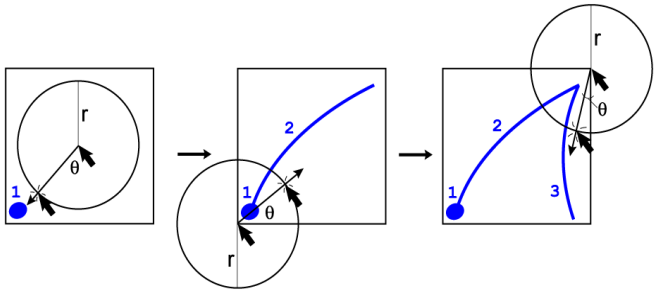

Figure 3. In pulsing to corners, users are performing *goal crossings*. This figure shows three crossings for "a". The first crossing determines the gesture's initial corner.

The fundamental concept underlying Trackball EdgeWrite is *crossing* [1] (Figure 3). Goal crossing contrasts with *pointing*, for which one must acquire an area target and remain within it. With crossing, one must only pass a goal line—like a football player scoring a touchdown—regardless of how fast one moves. The demand for accuracy can be lessened with crossing instead of pointing because one does not have to remain inside a target [3].

In Trackball EdgeWrite, when a user pulses the trackball towards a corner, he or she is performing a crossing task. Although the mouse cursor is invisible, it is actually crossing the circumference of a circle. The angle at which this occurs determines the intended corner. The benefit of this type of motion is that no targets must be pointed at and the mouse can move arbitrarily fast—all that matters are the *angles* formed by pulses on the trackball.

In Trackball EdgeWrite, to switch from mousing to writing, users can "capture" the mouse by pressing a button. Alternatively, a user can dwell in the corner of the desktop screen. When captured, the mouse cursor becomes invisible and subsequent trackball motion creates strokes within a revealed EdgeWrite square, sending text to the active application (e.g. Microsoft Word). When the user is done writing, a button press or dedicated stroke releases the cursor to resume mousing. Note that no buttons are necessary for writing—an advantage for motor-impaired users.

3.2 Word Completion in Trackball EdgeWrite

3.2.1 Interaction Design

A key aspect of our design is that character-level stroking remains unchanged from the prior version of Trackball EdgeWrite. This is important for two reasons: (1) it allows current users to remain effective with the software, and (2) it allows users to gradually ramp up to using word completions at their own pace.

When a user strokes, candidate words are shown at the four corners of the EdgeWrite square (Figure 4). In order to provide completions, the current stroke is recognized after each corner is entered. We call this *continuous recognition feedback*, which also displays the

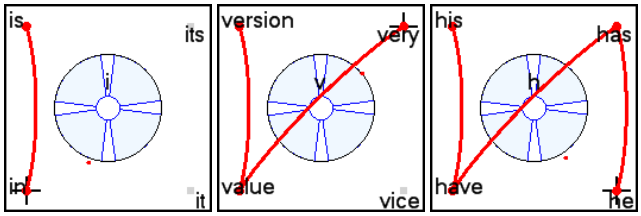

Figure 4. As the user makes an "h", the system recognizes an "i" and "v" along the way, offering English frequency-based completions as each corner is entered.

current stroke result in the center of the EdgeWrite square. Thus, the user knows what his or her stroke will produce before the stroke is segmented by a slight pause. If the user slips, he or she can simply restart the stroke before segmenting using a feature called *non-recognition retry* [30].

After a stroke is segmented and a letter is produced, the user can continue stroking letters or, alternatively, make a short gesture to select a word. This gesture is a singular motion from the center of the EdgeWrite square to the corner containing the desired word.

In the event of an erroneous completion, the user can make a backspace stroke along the bottom of the square, undoing the selection and restoring the completions as they appeared before. This makes completions quickly undoable.

An important aspect of this design is that the same completion is always shown in the same corner for the same prefix. This is because completions are based only on English word frequencies, not on context. This consistency is important for enabling users to rely on the positions of words. For example, after stroking a "t", the word "the" is always shown in the lower-right corner (Figure 1). Thus, users can come to rely on the position of "the" and stroke it by feel rather than by sight. Zipf's law for language says that a small percentage of words make up a large percentage of written language [34,35], so an increase in the entry rates of a few words can produce an overall speed gain. The consistency of word positions also may reduce cognitive load as motor performance comes to dominate.

In addition to showing frequency-based word completions, Trackball EdgeWrite also shows context-dependent word predictions after a word ends. Word predictions are, by definition, contextual and thus cannot be stroked by feel.

3.2.2 Language Coverage

Trackball EdgeWrite's design for stroke-based word completion avoids high perceptual search times by showing only four words at a time, generally less than most word completion systems [16]. But how useful are only four words? To answer this question, we wrote a computer program to calculate the amount of language coverage obtained for 1-5 letter prefixes showing only four frequency-based completions per letter (Figure 5, next page). We used Kucera-Francis frequencies for the 17,805 most common English words [17]. According to the graph, users have a 49.0% chance of seeing their intended word after just one letter! After two letters, this climbs to 70.8%. After three, it's 89.3%. This is the Zipf's law effect [35].

As explained elsewhere [29], one can achieve a slightly higher language coverage by not re-showing the same word completions once they have been shown for a given word being entered. For example, when "t" is written, "the" is one possible completion. If an

"h" is written next, should "the" be re-shown? Or, since the user did not select "the", should a different word be shown in its place? In Trackball EdgeWrite, we implement the latter because users sometimes miss the initial appearance of the word they want, entering more letters than necessary. If completions are not re-shown, then this behavior costs them their chance to select their desired word.

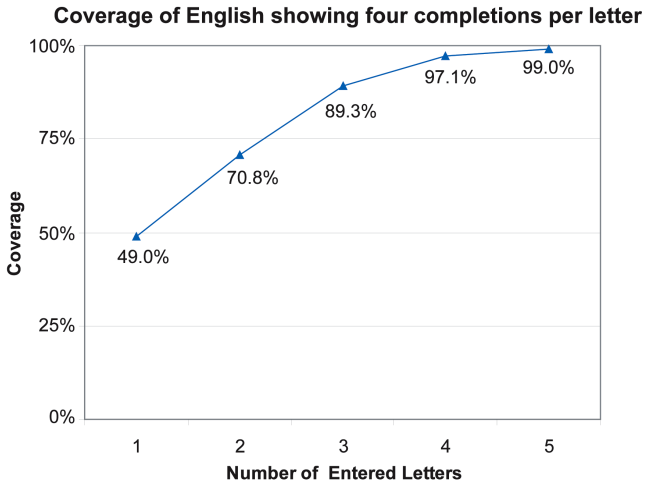

Figure 5. Coverage of the 17,805 most common English words [17] based on 1-5 letter prefixes and four frequency-based completions shown per entered letter.

3.3 Implementation

Our word prediction and completion system has four main components: (1) a vocabulary list of words and frequencies, (2) an optional user-defined vocabulary list, (3) a trigram list with trigram frequencies, and (4) an adaptive bigram cache that stores a user's words at runtime. The first and second provide "fixed" frequency-based word completions as words are being made. The third and fourth provide context-dependent word predictions after a word has been completed (i.e. after a SPACE has been entered).

The vocabulary list is stored in an alphabetically sorted array enabling binary search for fast lookups. Each array slot contains a word string and the word's frequency count. This is all the data necessary for fixed frequency-based word completions. Also in each slot is a hash table whose keys are word indices and whose values are a list of word indices. The slot's word string represents the first word of a trigram, its hash table keys represent second words, and its hash table list values represent third words. These data structures allow fast lookups for both fixed completions and context-dependent predictions.

When a letter is entered, words that match the current prefix are gathered from the vocabulary list. If a user-defined vocabulary list is loaded, its words with matching prefixes are also gathered. These words are then sorted in a separate list according to their frequencies. The top four words are then offered as completions. Since frequencies are based on English, these four completions will always be the same for a given prefix.

When four frequency-based words are retrieved from the language model, they are assigned to corners such that the highest priority word is given the corner in which the current stroke resides. The two adjacent corners receive the next two words, and the lowest priority word is placed at the diagonal away from the stroke's current corner. Once a word has been shown, it is stored in a hash table along with its corner and a *half-life*. If a word is shown again, it will be shown in the same corner as it was before. If the word goes unused for awhile, it will "decay" and be eligible for reassignment. If a collision occurs with two words vying for the same corner, the highest priority word wins.

When a SPACE is entered, context-dependent predictions are offered. The most recent two words are used to look up possible third word predictions. The first word is found in the vocabulary array using binary search. The second word's index, which was found when the word was entered, is hashed upon in the first word's hash table. The value returned, if any, is a list of possible third words. The top four are shown as predictions.

Predictions also come from an adaptive bigram cache. The cache holds recent bigrams so that when a user enters a previously used word, words that followed it can be offered as predictions. The cache is a list maintained in priority order such that when a new bigram is entered or an old bigram reused, it is placed at the top. Unlike the trigrams, the adaptive bigram cache accommodates out-of-vocabulary words, enabling the prediction of last names from first names, etc.

The English vocabulary list and trigrams were built by parsing 850MB of news articles from the Wall Street Journal, Ziff Davis, Los Angeles Times, and Associated Press. This parsing was carried out with the CMU-Cambridge Statistical Language Modeling toolkit [6]. Our own custom parsers then pared down the toolkit's results, keeping 20,000 of the most common words, and only trigrams that occurred 20+ times. After certain abbreviations were removed, the result was a 258KB vocabulary list of 19,122 words with frequency counts totaling 132,701,943. The maximum frequency count was for the word "the" at 7,686,122, or 5.79%. Our trigram list is 10.6MB and contains 517,988 trigrams with frequency counts totaling 40,230,622. The maximum frequency count is for the trigram "the United States" at 46,947, or 0.12%. Although we used news articles, our procedure could easily be re-run over other corpora (e.g. email).

Our stroke-based word prediction and completion system is part of an EdgeWrite library (DLL) that can be used with any .NET language. The library is built in C# and provides full EdgeWrite text entry in a few lines of code. Its API comes fully documented and is available for free at http://www.edgewrite.com/dev.html.

3.4 Theoretical Model

In our original discussion of Trackball EdgeWrite [30], we calculated a theoretical upper-bound speed based on the Steering law [1]. Using Fitts' coefficients based on prior studies, we calculated "perfect" character entry in Trackball EdgeWrite to be 23.1 WPM. Although this speed is probably unachievable, it is reasonable as an upper-bound in light of expert speeds with other unistroke systems [20].

We now extend this theoretical model to incorporate frequency-based word completions. Using the same Fitts' coefficients and formulae for calculating individual character speeds as before [30], we wrote a computer program to calculate WPM assuming that each completion is selected when it appears. We did this for all words in Trackball EdgeWrite's list of 19,122 words, a list large enough to contain most words used in everyday English.

We can calculate the speed S_{cps} for our corpus using Equation 1:

$$S_{cps} = \sum_{w \in C} \left(\frac{|w|+1}{T_w} \times F_w \right) \times 1000 \quad (1)$$

Here, S_{cps} is the weighted speed of text entry in characters per second (CPS), w is a word in corpus C with length $|w|$, T_w is the time to write word w in milliseconds (ms), and F_w is the frequency of word w such that $\Sigma F_w = 1.00$. The "+1" in the numerator is for the space that is added after a completion is selected, and the "×1000" converts from characters per ms (CPMS) to CPS.

To calculate T_w in ms for each word in the corpus, we need to calculate the time T_ℓ to perform each letter $\ell \in w_p$, where w_p is the minimum prefix that will show w as a completion ($1 \le |w_p| \le |w|$). To this we add T_{select}, the time to select the completion itself (Equation 2). Note that part of the time included in T_ℓ and T_{select} is τ, the segmentation time after a letter or completion is made. As in our prior model, we use τ = 150 ms. Readers interested in the calculation of T_ℓ itself are directed to the prior model [30].

$$T_w = \left(\sum_{\ell \in w_p} T_\ell \right) + T_{select} \quad (2)$$

For words which themselves are prefixes of at least four other more common words (e.g. "ad"), there is no such w_p that will show w as a completion. For these words, w must be entered fully along with a trailing space, which is modeled by Equation 3:

$$T_w = \left(\sum_{\ell \in w} T_\ell \right) + T_{space} \quad (3)$$

To convert S_{cps} in Equation 1 from CPS to WPM, we use the standard definition of 5 characters per word:

$$S_{wpm} = S_{cps} \times \frac{60 \text{ sec}}{1 \text{ min}} \times \frac{1 \text{ word}}{5 \text{ chars}} \quad (4)$$

Using Equations 1-4, our model yields an upper-bound text entry rate of 52.5 WPM. This is 227% faster than the 23.1 WPM obtainable with only character-level strokes. Like before, this result is unachievable by a real user. It represents *perfect* entry, lacking considerations for hesitation, cognitive processes, visual search, slips, or mistakes. Still, it is useful as an upper-bound for theoretical comparisons with prior models.

For a better estimate, we can enrich our model by adding a term for visual search time based on the Hick-Hyman law [12,13]. This term T_n is added after the entry of every letter ℓ and represents the time it takes for a user to find their word amidst n choices, where n is the number of completions offered for the current prefix ($0 \le n \le 4$). Using the rationale from [26], our formula for T_n in ms is:

$$T_n = 0.2 \times \log_2(n) \times 1000 \quad (5)$$

Incorporating the Hick-Hyman law, Equations 2-3 become:

$$T_w = \left(\sum_{\ell \in w_p} (T_\ell + T_n) \right) + T_{select} \quad (6)$$

$$T_w = \left(\sum_{\ell \in w} (T_\ell + T_n) \right) + T_{space} \quad (7)$$

Using Equations 5-7, our result drops 36.2% from 52.5 WPM to 33.5 WPM. This is a more realistic result. Note, however, that even with the addition of visual search time, this result still represents perfect entry. This result is 45.0% faster than the 23.1 WPM result from the character-level model [30].

A limitation of this model is that it does not account for word prediction. However, modeling word prediction is more difficult because it depends on context, including the user's adaptive cache of recent words. Such a model is therefore beyond the current scope.

4. EMPIRICAL VALIDATION

In order to empirically test stroke-based word prediction and completion in Trackball EdgeWrite, we conducted two evaluations with a 15-year trackball veteran with a spinal cord injury. The first was a comparison to the WiViK on-screen keyboard. The second was an analysis of our subject's log files over two months of intermittent use.

4.1 Comparison to On-screen Keyboard

4.1.1 Subject

Our subject, who we will call "Jim," has had a spinal cord injury for over 15 years and has used a trackball for about as long. Although he also uses voice recognition, he is often dissatisfied with it and, until recently, has relied on an on-screen keyboard as a complementary method. His on-screen keyboard of choice has been the Microsoft Accessibility Keyboard, which does not have word prediction or completion. However, the keyboard *does* have useful visual feedback when hovering over keys, which Jim relies on to enter text since he cannot reliably click. About six months ago, Jim stopped using on-screen keyboards in favor of Trackball EdgeWrite, even before it had word completion capabilities.

Jim's *best* prior performance with character-level Trackball EdgeWrite was 8.25 WPM with 6.24% total errors, and with his on-screen keyboard was 6.90 WPM with 6.60% total errors. However, these entry rates seemed to be plateaus. Our goal was therefore to see how Jim's speeds would compare when these methods were given word prediction and completion.

4.1.2 Apparatus

Since Jim's preferred on-screen keyboard does not have word prediction, we configured the popular WiViK on-screen keyboard (http://www.wivik.com) to match Jim's desired settings: 550 ms dwell, about 650×250 pixels in size, and no "dead space" between keys. For word prediction and completion, WiViK uses a program called *WordQ*, which we loaded with the "US Advanced" database containing 15,000 words. WiViK shows a vertical 6-item word list to the left of the keyboard. The same action for selecting a key selects a word in the word list—in Jim's case, by hovering for 550 ms.

Jim keeps his monitor set to 800×600 resolution. Test phrases [21] were randomly presented using the *TextTest* program, which creates XML log files that can be analyzed with the *StreamAnalyzer* program [31]. StreamAnalyzer produces results according to the measures in [27,31].

4.1.3 Procedure

The study was a single-subject 2-factor design, with factors for *method* (WiViK, EdgeWrite) and *word prediction* (on, off). Jim did the word prediction versions second within both methods. A coin

toss determined that he used WiViK first. Thus, the order was: WiViK, EdgeWrite, WiViK+WP, EdgeWrite+WP. Jim entered 3 practice phrases and 8 test phrases in each condition. Each phrase was approximately 30 characters long.

4.1.4 Quantitative Results

Figure 6 shows Jim's speeds for the four conditions in the current study. It also shows Jim's prior peak speeds with his own on-screen keyboard and with character-level Trackball EdgeWrite. Note the substantial speedups of both methods due to word prediction and completion. Figure 7 shows corresponding total error rates. However, because Jim fixed almost every error during entry, these total error rates are really just corrected error rates [27]. Corrected errors, which slow entry rates, are of less concern than *uncorrected* errors, which are at odds with speed. Thus, Trackball EdgeWrite is producing similarly accurate text in a tad less time, albeit with more errors made (and fixed) along the way.

A Wilcoxon sign test for speed is not significant (z=3.0, p=0.25). However, the general trend is in favor of Trackball EdgeWrite. This advantage is only slight for the current study, however, probably because WiViK is superior to Jim's preferred keyboard, the Microsoft Accessibility Keyboard, even though WiViK was configured with Jim's usual settings.

A Wicoxon sign test for total errors is also not significant (z=2.0, p=0.50). However, both methods were producing error-free text in the end, since uncorrected errors for both methods were ~0.00%. It is interesting that Trackball EdgeWrite's errors were low in the current study even *without* word completion, probably because Jim has had more practice since his prior peak performance.

It is worth noting that the speed of WiViK improved 32.0% with word prediction compared to without. This confirms prior results [4] and highlights the strength of WiViK's commercial word prediction and completion technology.

Taken together, these results show a 46.5% increase in speed and a 36.7% decrease in errors for word-level Trackball EdgeWrite compared to Jim's prior peak performance with character-level Trackball EdgeWrite. The results also show that word-level Trackball EdgeWrite is 75.2% faster and 40.2% more accurate than Jim's prior peak performance with his preferred on-screen keyboard. Finally, the results show that word-level Trackball EdgeWrite is competitive with a major commercial product, the WiViK on-screen keyboard with word prediction and completion.

4.1.5 Qualitative Results

We asked Jim to describe his experience of each of the four conditions in his own words:

- *WiViK*: "[Y]ou are constantly either scribbling around so you don't accidentally trigger the wrong letter, checking to see if you typed the right thing, or looking for the next key to hover over. Too much work both mentally and visually."
- *WiViK+WP*: "Somewhat of a relief to hover over large words but it just increased the amount of mental and visual work required. [It's] one more section of the screen you need to scan constantly. Only thing is, I wish EdgeWrite had its vocabulary."
- *EdgeWrite*: "EdgeWrite without word prediction is like using a 286 or something. It's much better than a keyboard or an on-screen keyboard, but the ultimate is when you can flick the cursor into a corner and just pop the rest of the word in."
- *EdgeWrite+WP*: "The best thing about EW is there is no eye strain or constant scanning between programs, letters, words, etc. The word choices are right there where your eyes already are. It actually helps you stay focused on what you're writing."

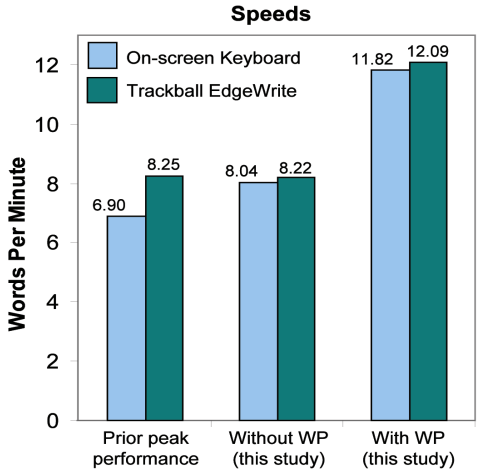

Figure 6. On-screen keyboard and Trackball EdgeWrite speeds. Higher values are better.

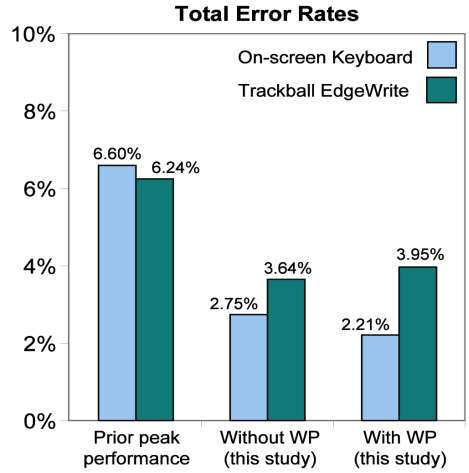

Figure 7. On-screen keyboard and Trackball EdgeWrite total error rates. Lower values are better.

Jim's sentiments confirm what prior studies of on-screen keyboards have found: that they are exceedingly tedious and visually intense [4]. Although word prediction and completion improved WiViK's speed by 32.0%, it did not resolve these drawbacks. Trackball EdgeWrite, on the other hand, proved to be just as fast but without the same visual tedium.

4.2 Log File Analysis of Extended Use

A single-session lab study allows us to formally quantify speed and errors, but it is over the long-term that we hope Trackball EdgeWrite will be useful. Indeed, prior studies of word prediction systems have shown that long-term use is critical for accurate evaluations [22]. Furthermore, the design of our stroke-based word completion system supports *gradual* adoption as users familiarize themselves with the consistent positions of words.

Although log files do not enable us to rigorously quantify speed and accuracy, they do allow us to measure the stroke savings gained by using word completion. We can also look at the number of completions undone as an approximation of selection accuracy, and compare this to the number of letters undone (backspaced).

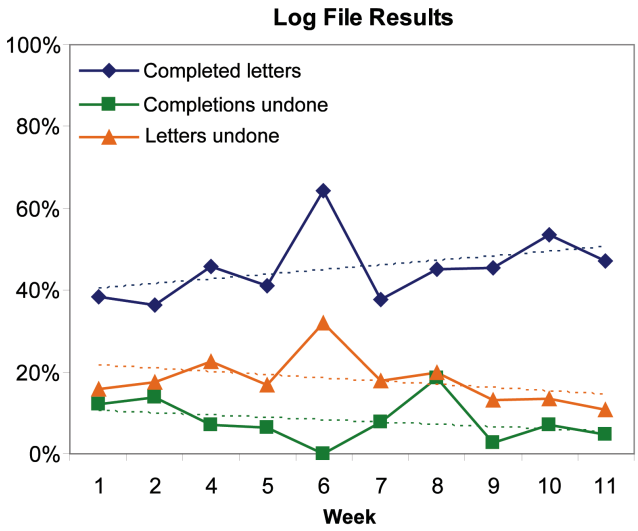

Figure 8. Results over 11 weeks from extended use showing usage of word completion and backspace. Week 3 is omitted because Jim did not use his computer.

Figure 8 shows these quantities graphed over two months of Jim's intermittent use. It represents 897.52 hours of software running-time for 13,288 total strokes. Of these, 8774 were character strokes and 2201 were word-selection strokes. In all, 15,629 characters were entered, 6855 of which were from completions.

The top line (blue) shows the percent of letters entered as predictions or completions. Without stroke-based word completion, these letters would all have to be entered in full. The weighted mean over all weeks is 43.9%. The spike in week 6 is an outlier due to a week of relative inactivity. Only 70 letters were entered that week, compared to most weeks which saw 1500-3500 letters. A regression line shows this trend to be slightly increasing.

It is interesting that the 43.9% savings shown in Figure 8 is about the same as the 46.5% speedup shown in Figure 6. That is, the stroke savings more or less translate to speed gains. This suggests that the perceptual, cognitive, and motor costs of stroke-based word completion are not overly taxing, as often has been the case with prior word prediction systems.

The bottom line (green) is the percentage of word completions undone. The weighted mean over all weeks is 7.7%. As an indicator of completion errors, this value is probably high, since users may undo selected completions for reasons other than errors (i.e. as a result of changing what they intended to write). A regression line shows this trend to be slightly decreasing.

For comparisons, the percentage of letters undone (backspaced) is shown as the middle line (orange). The weighted mean for undone letters is 16.5%. Although this value is high, it is not surprising in light of previous results indicating that BACKSPACE is the second most common keystroke in desktop text entry [20]. A regression line shows this trend to be slightly decreasing.

Across all weeks, the average number of characters entered per completion was 3.11. Thus, with a simple "pulse" into one of four corners, users avoid entering over 3 more characters for every word they write.

5. FUTURE WORK

Although stroke-based word completion substantially improves the speed of Trackball EdgeWrite, it could still be improved upon. One of Jim's quotes in section 4.1.5 indicated that he preferred the words offered by WiViK to those offered by Trackball EdgeWrite. We could recreate EdgeWrite's language models using sources other than newspaper articles, perhaps including some of Jim's own texts. The culmination of this idea would be to provide the end-user with an interface to incorporate their *own* texts into EdgeWrite's language models.

An obvious next step is to run a study with a larger number of users. At this stage, we preferred to study one subject in-depth to verify the usefulness and usability of our technique. Now we believe it is ready for a broader assessment. We have one user in Sweden who is already using Trackball EdgeWrite, and we intend to formally evaluate her speed and accuracy.

We are also studying how Trackball EdgeWrite performs with an isometric joystick embedded in a mobile phone [5]. Like trackballs, isometric joysticks have no notion of position, so the same software works without modification. Initial results suggest that stroke-based word completion is competitive with T9 (http://www.tegic.com) on mobile phones, allowing able-bodied users to reach speeds near 20 WPM.

6. CONCLUSION

We have shown that Trackball EdgeWrite greatly benefits in terms of speed and accuracy from having stroke-based word prediction and completion. Our subject's best prior performance with character-level Trackball EdgeWrite was both slower and less accurate than his performance with the new word-level version. We also demonstrated that Trackball EdgeWrite rivals a major commercial on-screen keyboard. Our study confirms that Trackball EdgeWrite is just as fast using word prediction and completion, and that it is less visually tedious. Although Trackball EdgeWrite is more error prone *during* entry, it produces error-free text in the same amount of time due to efficient error correction. These results could be important for motor-impaired users who wish to *write* with their trackballs instead of hunting and pecking.

7. ACKNOWLEDGEMENTS

The authors thank Richard Simpson and Duen Horng Chau. This work was supported in part by Microsoft, General Motors, and the National Science Foundation under grant UA-0308065. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the National Science Foundation.

8. REFERENCES

[1] Accot, J. and Zhai, S. (1997) Beyond Fitts' law: Models for trajectory-based HCI tasks. *Proceedings CHI '97*. New York: ACM Press, 295-302.

[2] Accot, J. and Zhai, S. (1999) Performance evaluation of input devices in trajectory-based tasks: An application of the Steering Law. *Proceedings CHI '99*. New York: ACM Press, 466-472.

[3] Accot, J. and Zhai, S. (2002) More than dotting the i's: Foundations for crossing-based interfaces. *Proceedings CHI '02*. New York: ACM Press, 73-80.

[4] Anson, D. K., Moist, P., Przywara, M., Wells, H., Saylor, H. and Maxime, H. (2005) The effects of word completion and word prediction on typing rates using on-screen keyboards. *Proceedings RESNA '05*. Arlington, Virginia: RESNA Press.

[5] Chau, D. H., Wobbrock, J. O., Myers, B. A. and Rothrock, B. (2006) Integrating isometric joysticks into mobile phones for text entry. *Extended Abstracts CHI '06*. New York: ACM Press, 640-645.

[6] Clarkson, P. R. and Rosenfeld, R. (1997) Statistical language modeling using the CMU-Cambridge toolkit. *Proceedings of Eurospeech '97*, 2707-2710.

[7] Cook, A. M. and Hussey, S. M. (2001) *Assistive Technologies: Principles and Practice*, 2nd ed. St. Louis: Mosby Press.

[8] Dawe, M. (2006) Desperately seeking simplicity: How young adults with cognitive disabilities and their families adopt assistive technologies. *Proceedings CHI '06*. New York: ACM Press, 1143-1152.

[9] Fichten, C. S., Barile, M., Asuncion, J. V. and Fossey, M. E. (2000) What government, agencies, and organizations can do to improve access to computers for postsecondary students with disabilities: Recommendations based on Canadian empirical data. *International Journal of Rehabilitation Research 23* (3), 191-199.

[10] Fuhrer, C. S. and Fridie, S. E. (2001) There's a mouse out there for everyone. *Proceedings CSUN '01*. California State University Northridge.

[11] Goodenough-Trepagnier, C., Rosen, M. J. and Galdieri, B. (1986) Word menu reduces communication rate. *Proceedings RESNA '86*. Arlington, Virginia: RESNA Press, 354-356.

[12] Hick, W. E. (1952) On the rate of gain of information. *Quarterly Journal of Experimental Psychology 4*, 11-26.

[13] Hyman, R. (1953) Stimulus information as a determinant of reaction time. *Journal of Experimental Psychology 45* (3), 188-196.

[14] Isokoski, P. and Raisamo, R. (2000) Device independent text input: A rationale and an example. *Proceedings AVI '00*. New York: ACM Press, 76-83.

[15] Koester, H. H. (2003) Abandonment of speech recognition by new users. *Proceedings RESNA '03*. Arlington, Virginia: RESNA Press.

[16] Koester, H. H. and Levine, S. P. (1996) Effect of a word prediction feature on user performance. *Augmentative and Alternative Communication 12* (3), 155-168.

[17] Kucera, H. and Francis, W. N. (1967) *Computational Analysis of Present-Day American English*. Providence, Rhode Island: Brown University Press.

[18] MacKenzie, I. S., Kauppinen, T. and Silfverberg, M. (2001) Accuracy measures for evaluating computer pointing devices. *Proceedings CHI '01*. New York: ACM Press, 9-16.

[19] MacKenzie, I. S., Sellen, A. and Buxton, W. (1991) A comparison of input devices in elemental pointing and dragging tasks. *Proceedings CHI '91*. New York: ACM Press, 161-166.

[20] MacKenzie, I. S. and Soukoreff, R. W. (2002) Text entry for mobile computing: Models and methods, theory and practice. *Human-Computer Interaction 17* (2), 147-198.

[21] MacKenzie, I. S. and Soukoreff, R. W. (2003) Phrase sets for evaluating text entry techniques. *Extended Abstracts CHI '03*. New York: ACM Press, 754-755.

[22] Magnuson, T. and Hunnicutt, S. (2002) Measuring the effectiveness of word prediction: The advantage of long-term use. *Speech, Music and Hearing 43*, 57-67.

[23] Mankoff, J. and Abowd, G. D. (1998) Cirrin: A word-level unistroke keyboard for pen input. *Proceedings UIST '98*. New York: ACM Press, 213-214.

[24] Perlin, K. (1998) Quikwriting: Continuous stylus-based text entry. *Proceedings UIST '98*. New York: ACM Press, 215-216.

[25] Soede, M. and Foulds, R. A. (1986) Dilemma of prediction in communication aids. *Proceedings RESNA '86*. Arlington, Virginia: RESNA Press, 357-359.

[26] Soukoreff, R. W. and MacKenzie, I. S. (1995) Theoretical upper and lower bounds on typing speed using a stylus and soft keyboard. *Behaviour and Information Technology 14* (6), 370-379.

[27] Soukoreff, R. W. and MacKenzie, I. S. (2003) Metrics for text entry research: An evaluation of MSD and KSPC, and a new unified error metric. *Proceedings CHI '03*. New York: ACM Press, 113-120.

[28] Ward, D. J., Blackwell, A. F. and MacKay, D. J. C. (2000) Dasher—A data entry interface using continuous gestures and language models. *Proceedings UIST '00*. New York: ACM Press, 129-137.

[29] Wobbrock, J. O. and Myers, B. A. (2006) In-stroke word completion. *Proceedings UIST '06*. New York: ACM Press. To appear.

[30] Wobbrock, J. O. and Myers, B. A. (2006) Trackball text entry for people with motor impairments. *Proceedings CHI '06*. New York: ACM Press, 479-488.

[31] Wobbrock, J. O. and Myers, B. A. (2007) Analyzing the input stream for character-level errors in unconstrained text entry evaluations. *Transactions on Computer-Human Interaction (TOCHI)*. To appear.

[32] Wobbrock, J. O., Myers, B. A. and Kembel, J. A. (2003) EdgeWrite: A stylus-based text entry method designed for high accuracy and stability of motion. *Proceedings UIST '03*. New York: ACM Press, 61-70.

[33] Wu, T.-F., Wang, H.-P. and Chen, M. C. (2005) Enabling computer access for children with cerebral palsy. *Proceedings HCI Int'l '05*. Mahwah, New Jersey: Lawrence Erlbaum. On proceedings CD.

[34] Zhai, S. and Kristensson, P. (2003) Shorthand writing on stylus keyboard. *Proceedings CHI '03*. New York: ACM Press, 97-104.

[35] Zipf, G. (1932) *Selective Studies and the Principle of Relative Frequency in Language*. Cambridge, Massachusettes: MIT Press.