Taming Wild Behavior: The Input Observer for Obtaining Text Entry and Mouse Pointing Measures from Everyday Computer Use

Abigail C. Evans and Jacob O. Wobbrock
The Information School | DUB Group
University of Washington
Seattle, WA 98195 USA
{abievans, wobbrock}@uw.edu

ABSTRACT

We present the *Input Observer*, a tool that can run quietly in the background of users' computers and measure their text entry and mouse pointing performance from everyday use. In lab studies, participants are presented with prescribed tasks, enabling easy identification of speeds and errors. In everyday use, no such prescriptions exist. We devised novel algorithms to segment text entry and mouse pointing input streams into "trials." We are the first to measure errors for unprescribed text entry and mouse pointing. To measure errors, we utilize web search engines, adaptive offline dictionaries, an Automation API, and crowdsourcing. Capturing errors allows us to employ Crossman's (1957) speed-accuracy normalization when calculating Fitts' law throughputs. To validate the Input Observer, we compared its measures from 12 participants over a week of computer use to the same participants' results from a lab study. Overall, in the lab and field, average text entry speeds were 74.47 WPM and 80.59 WPM, respectively. Average uncorrected error rates were near zero, at 0.12% and 0.28%. For mouse pointing, average movement times were 971 ms and 870 ms. Average pointing error rates were 4.42% and 4.66%. Average throughputs were 3.48 bits/s and 3.45 bits/s. Device makers, researchers, and assistive technology specialists may benefit from measures of everyday use.

Author Keywords: Text entry, mouse pointing, everyday, "in the wild," field studies, human performance, Fitts' law.

ACM Classification Keywords: H.5.2 [Information Interfaces and Presentation]: User interfaces—*evaluation / methodology*.

INTRODUCTION

There is a growing desire in human-computer interaction (HCI) to "break from the lab," *i.e.*, to understand users' behavior, performance, attitudes, emotions, *etc.* in everyday computer use [5,6,25,36,37]. Methods like ethnography and diary studies have supported this trend, as have novel tools for experience sampling (*e.g.*, [17]) and behavior sensing and recognition (*e.g.*, [8]). However, while these methods and tools are well-suited to understanding some aspects of human behavior, they do not suffice for understanding human performance with *input techniques*, which, for our purposes, are text entry and mouse pointing techniques. Understanding human performance with input techniques has traditionally required controlled lab studies for accurate and rigorous quantification of speeds and errors (*e.g.*, [42,43]). Hence, "breaking from the lab" has not occurred for input research as it has for other areas of HCI.

CHI'12, May 5–10, 2012, Austin, Texas, USA.

While input-oriented field studies are seen occasionally (*e.g.*, [7,22]), basic logging can only produce counts, proportions, and durations-of-use. In the wild, calculating text entry speeds in words per minute, or pointing speeds in milliseconds, requires parsing out portions of data input streams. Calculating error rates requires knowing what users *intended* to do. Such a requirement can feel frighteningly close to a need for mind-reading. Unsurprisingly, prior studies of pointing "in the wild" [7,22] ignored error rates.

To obtain human performance measures commonly used in the lab from everyday computer use, we created the *Input Observer* (*cf.* [13]). The Input Observer runs on Windows and measures text entry and mouse pointing by: (1) observing keyboard and mouse input streams using low-level hooks; (2) extracting "trials" where input behavior is directed, contiguous, and similar to behavior in the lab; (3) measuring speeds from "trial"-starts to "trial"-ends; (4) measuring errors by, (*i*) in the case of text, consulting adaptive offline dictionaries and the Bing search engine, and (*ii*) in the case of mouse pointing, consulting the Windows Automation API and crowdsourced results with Mechanical Turk in which Turkers are asked to identify intended targets; and (5) measuring Fitts' law *throughput* [15,27,32,43,50]. Crucially, our ability to measure pointing errors enables us to employ Crossman's [9] speed-accuracy correction based on the spread of hits around targets at the same distance (A) and of the same size (W). As pointing in the wild occurs without any notion of $A \times W$ conditions, we employ the *G-means* [19] and *k-means++* [2] clustering algorithms to extract *post hoc* $A \times W$ "conditions," similar to those used lab experiments [43]. Without the use of Crossman's correction, throughputs cannot be normalized to make comparable, *e.g.*, fast-but-sloppy users and slow-and-careful users [16,32,43,49,51].

We envision at least three types of potential users of the Input Observer: (*i*) assistive technology specialists seeking to gain quantitative insights into clients' performance with devices in home settings, as such specialists have little time to match clients to the best possible devices [12]; (*ii*) device manufacturers seeking to understand, through extended field testing, whether new prototypes outperform existing devices; and (*iii*) researchers seeking to add an *in situ* component to their understanding of new input techniques and devices. Researchers may also seek quantitative models for parameterizing an automatic user interface generator such as *SUPPLE* [18] to create "ability-based user interfaces" tailored to users' motor skills without requiring a controlled test. In general, the Input Observer can be a tool for supporting various aspects of *ability-based design* [48].

To validate the Input Observer, we compared its measurements of 12 participants over a week of everyday computer use to the performance by the same participants in controlled lab studies. Overall, in the lab and field, average text entry speeds were 74.47 WPM and 80.59 WPM, respectively. Average uncorrected error rates [42] were near zero, at 0.12% and 0.28%. For mouse pointing, average movement times were 971 ms and 870 ms. Average pointing error rates were 4.42% and 4.66%. Average throughputs were 3.48 bits/s and 3.45 bits/s.

The primary contribution of this paper is the development of a new tool for measuring text entry and mouse pointing performance from everyday computer use based on two novel algorithms for extracting lab-like "trials" from undifferentiated text entry and mouse pointing input streams. Secondary contributions are: (1) an approach to measuring text entry errors using the web; (2) an approach to measuring pointing errors using crowdsourcing; and, (3) a method for applying Crossman's [9] speed-accuracy normalization to "wild" pointing data without prescribed target distances (A) or target sizes (W).

RELATED WORK

Chapuis *et al.* [7] conducted a study of Fitts' law "in the wild," demonstrating that Fitts' law holds in the field as long as steps are taken to reduce the "noise" present in field data. Chapuis *et al.* segmented pointing movements using pauses, treated all movements as accurate, used the Accessibility API to extract 22% of possible target bounds, and averaged over many individual pointing trials using an arbitrary number of quantiles based on index-of-difficulty values. Hurst *et al.* [22] conducted a similar study, although with people of varying pointing abilities. Both studies found that pointing "in the wild" differs from pointing in the lab in important ways. Unlike the current work, however, these studies did not measure errors, extract *post hoc* $A \times W$ "conditions," apply Crossman's correction [9] in Fitts' law, or produce a reusable tool for others to employ. They also did not consider text entry at all.

In other work, Hurst *et al.* [21] were able to distinguish between novice and skilled users by applying learned statistical models to mouse data from a specially instrumented application. Although Hurst *et al.*'s participants had no task model and the data was gathered "in the wild," data was only gathered from one application built by the researchers. In contrast, the Input Observer collects mouse and text entry data directly from the operating system and is therefore application-agnostic.

We are not the first to attempt to identify real-world targets. Hurst *et al.* [23] used a combination of the Accessibility API, machine learning, and computer vision to get target information. The Accessibility API alone enabled the researchers to find 74% of on-screen targets, but when it was combined with machine learning and computer vision, 84% of targets were discovered. For a different approach, Dixon and Fogarty [11] used pixel-based matching methods to identify targets. We use yet another approach, namely a combination of the Windows Automation API[1] and crowdsourcing on Mechanical Turk to identify targets. An advantage of using crowdsourcing is that it enables us to identify pointing errors for the first time.

MEASURING "WILD" TEXT ENTRY

The review above makes it clear that although pointing performance outside the lab has been touched on by prior work, everyday text entry has been largely ignored. In this section, we describe the Input Observer's approach to measuring text entry performance beyond the lab.

Collecting Data beyond the Laboratory

In controlled studies of text entry performance, users transcribe presented phrases as "quickly and accurately as possible" [47]. Each phrase is considered a single trial. Transcribing presented phrases ensures that participants only need to *copy* text, not *compose* it, which would ruin experimental control, error measurement, and reproducibility [29]. Further, controlled text entry studies disallow the use of the mouse cursor or text cursor keys during entry, permitting *backspace* as the only mechanism for correction [47]. Doing so enables error rate calculation [42], but limits the ecological validity of lab studies.

Key performance measures calculated in controlled text entry studies are words per minute (WPM) and uncorrected error rates [42,47]. *Uncorrected errors* are those remaining in the final transcribed string. There are also *corrected errors*, which are any characters backspaced during entry, but these are of less interest because error correction takes time and is therefore subsumed in WPM [47] (*see* p. 56). The Input Observer is produces all of these measures, but without presenting phrases for transcription. Instead, the tool examines the text input stream and extracts phrases, or "trials," in which text entry is continuous.

Segmenting Text Entry "Trials" from the Input Stream

While the Input Observer runs on a participant's computer, it collects text entry data. However, the software does not

[1] The Automation API supersedes the Accessibility API. See http://msdn.microsoft.com/en-us/library/dd561918(v=vs.85).aspx.

log the text input stream in its entirety. Instead, the stream is segmented into "trials" as the participant types. Each "trial" has a start point—the first key-press—and an end point—the last key-press—before an identified *segmenting event*. In everyday text entry, finding the start and end points analogous to those from a controlled trial can be tricky. For segmenting events, we use the entry of end-of-sentence punctuation (*e.g.*, periods, exclamation points, and question marks), the ENTER key, and characters not appearing in the MacKenzie and Soukoreff text entry phrase set [30]. Successive capital letters and numbers are also segmenting events, as is any mouse movement.

Pauses are also used for segmentation. However, a single pause value is not sufficient to properly segment everyday text input streams into "trials" similar to those from lab studies. Users pause for different lengths of time while typing depending on whether they are typing letters or backspaces, or transitioning between the two. Empirically, we observed from 9 participants that two successive non-backspaces or backspaces were fastest (156 ms); a non-backspace following a backspace was next (247 ms); and a backspace following a non-backspace was slowest (465 ms). Adding 3 *SD* to each of these means gives us our three pause segmentation times: 1270 ms, 2085 ms, and 3215 ms.

With the criteria above, some segmented phrases can be very short, even a few characters. Such phrases result in unreliable and inaccurate measures; therefore, to be logged as "trials," segmented phrases must contain at least 24 characters. This length is 1 *SD* less than the mean length of phrases in the MacKenzie and Soukoreff phrase set [30].

Measuring text entry speed is straightforward once a "trial" is properly segmented [47]. However, text entry error rates are much more complicated. A source of complication is distinguishing text entry *errors* from *edits*, described next.

Distinguishing Errors from Edits

In a lab study, all backspaces can be regarded as error corrections because participants are attempting to match presented strings. Outside the lab, however, backspaces may correct errors, or they may indicate "changes of mind." We therefore must distinguish *errors* from *edits*, an issue that affects both corrected and uncorrected error rates. While backspaces from error corrections must remain in a "trial" to measure corrected error rates, backspaces from editing should not be included, as they do not reflect errors.

To distinguish errors from edits, backspaced text is compared to the text entered in its place, word by word. If users stop backspacing partway through a word, as in Figures 1a & 1c, the partial word is extended up to the nearest space to make a complete word. If the backspaced word is not the same as the word that replaced it, the Bing API's spell query[2] is used to identify errors in the backspaced word. If the spell query returns a suggested spelling for the backspaced word, the suggested word is compared to the word re-entered by the user. If the two words are the same, the original edits are considered *errors* (Figures 1a-b). If Bing has no suggestion, or the suggested word and the re-entered word are different, then the backspaces and subsequent entries are *edits*, and the phrase is segmented just before the first backspace (Figures 2c-d).

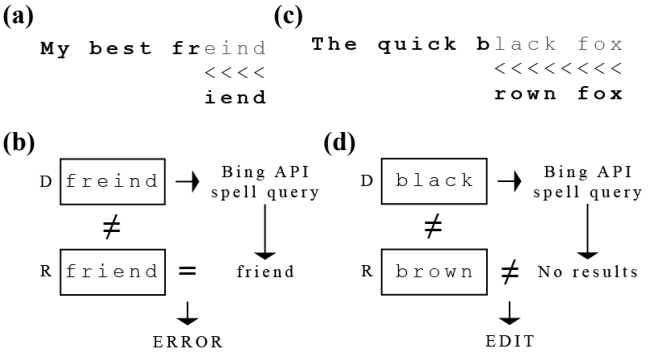

Figure 1. (a) A text entry input stream showing the correction of mistyped "freind" to "friend". **(b)** The mistyped "freind" is deemed "friend" by Bing, which matches the user's final word, so "freind" has two *errors*. **(c)** A text entry input stream showing a change of mind from "black" to "brown". **(d)** The backspaced "black" gives no spelling results from Bing, indicating it was not misspelled. As it does not equal "brown", the changes are called *edits* and no errors are counted.

Corrected Error Rate Calculation

Corrected errors are characters that are backspaced during entry and therefore do not remain in the transcribed string [42]. As described in the previous subsection, backspaces used to correct errors are distinguished from backspaces used to edit text. Therefore, when a user's log file is analyzed, the corrected error rate can be calculated simply from the backspaces recorded in the log.

Uncorrected Error Rate Calculation

In controlled text entry studies, uncorrected errors are calculated using the *minimum string distance* between the presented and transcribed strings [41,42]. The Input Observer has no presented strings, so uncorrected errors must be calculated from transcribed strings another way.

To measure uncorrected errors, each "trial" is broken into words by looking for spaces and between-word punctuation. Each word is checked against an offline lexicon containing ~80,000 words from the freely available Washington University in St. Louis *English Lexicon Project* [3]. If the word is found, it is considered correct.

If the word is not found, the Input Observer calls the Bing API's spell query. If the word contains an error recognized by the API, the query returns a suggested word (Figure 1b). In such cases, the word entered by the user is marked as containing one or more errors, and the suggested word from Bing is taken to be the intended word. The minimum string distance [41] between the entered word and the suggested word is calculated for the uncorrected error rate [42]. To reduce repeat queries to the Bing API, suggested words returned by Bing are added to the offline lexicon.

[2] http://msdn.microsoft.com/en-us/library/dd251056.aspx

When no suggested word is returned from a spell query to the Bing API, the word entered by the user is either error-free, or it contains an unrecognized error for which Bing cannot find an alternative. To distinguish between the two cases, an additional query is sent to the Bing API to define the word using the syntax `define:<word>`. If the query returns any results, then the word has a definition and is taken to be correct and added to the lexicon. If no definition is found, then the word is marked as containing an unknown error and cached for later. At the end of text entry data collection, the average minimum string distance from words containing known errors is applied to all cached words containing unknown errors.

Segmenting pauses frequently occur in the *middle* of words. Mid-word pauses can lead to partial words at the start and end of "trials," falsely inflating the uncorrected error rate. To address this issue, the initially-segmented phrase is maintained for the speed calculation but may be adjusted to complete partial words for the purpose of error-checking. If the last character before the pause is a letter or hyphen, it is assumed that the word was split by the pause and the first word of the next "trial" is adjusted to include the part of the word entered before the pause. Any errors that are found are only counted as part of the second "trial" to prevent errors from being counted twice.

Other segmenting events, such as mouse movements and cursor key-presses, can also occur in the middle of words. In these cases, the position of the text entry cursor may have changed since the last entered text so it is not possible to complete partial words. Instead, if an error is found in the first word of a "trial" following such a segmenting event, that word is omitted from the uncorrected error calculation.

As noted above, in lab studies, participants may only use backspace to correct errors [47]. In everyday use, however, users can employ several methods, including the mouse and cursor keys, to position the text cursor for error correction. To date, no theoretical breakthroughs have enabled the handling of the mouse or cursor keys in text entry error measurement. As a result, errors corrected using the mouse or cursor keys remain in segmented "trials" and falsely appear as uncorrected errors. To address this for cursor keys, phrases segmented by them are not included in the calculation of uncorrected or corrected error rates. We do not address this issue for mouse-based error correction.

Text Entry and Privacy

Clearly, the Input Observer's text entry measurement features introduce privacy concerns, as every extracted text "trial" is recorded in a log file. The Input Observer's minimum trial length of 24 characters ensures that usernames and passwords are not logged. However, longer phrases of text from personal communications still raise privacy concerns. We added an obfuscation feature that causes the Input Observer to log the letter "m" in place of actual text. In this case, the Input Observer still performs the above measurements on entered text, stores the results, and then logs only "m" characters. In addition, participants can turn off text entry logging at any time.

MEASURING "WILD" MOUSE POINTING

The pointing performance measures calculated by the Input Observer are time (ms), error rate (%), and throughput (bits/s), an important combined speed-accuracy measure of efficiency [15,27,32,43,50]. The Input Observer also generates MacKenzie *et al.*'s path analyses [28], although they are not reported here due to space constraints.

Unlike in controlled pointing studies, in everyday computer use, there are no defined trials or conditions. An important feature of the Input Observer is its ability to extract pointing "trials" that resemble pointing behavior in the lab and build *post hoc* "conditions" from myriad unsorted pointing events. The Input Observer is also the first to measure error rates for unprescribed pointing.

Segmenting Pointing "Trials" from the Input Stream

Rosenbaum [38] argues that the leading explanatory theory of Fitts' law is Meyer *et al.*'s [33] *optimized initial impulse model*. In this model, an aimed pointing attempt comprises a ballistic movement to the target vicinity and one or two optional corrective submovements for acquisition. The Input Observer examines mouse movements using this model to extract aimed pointing "trials." When the user signals the end of a pointing attempt with a click (Figure 2, F), the Input Observer moves through the movement backwards in time to find the "trial" start. This may be the first movement after a previous click, or it may be the first movement after the mouse velocity last fell to zero (Figure 2, A), whichever occurred closer in time to the current click. Prior to this scan, the mouse velocity is temporally resampled at 100 Hz and then smoothed using a Gaussian kernel filter with standard deviation parameter of 3 [14].

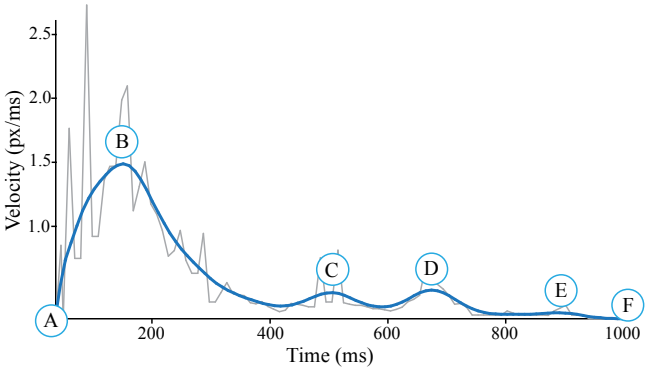

Figure 2. An actual velocity profile, recorded by the Input Observer, of a mouse movement that meets Meyer *et al.*'s [33] criteria for an aimed pointing movement.

With start and end points now identified, the smoothed velocity profile is scanned to find all local maxima and minima. The highest peak (Figure 2, B), representing the ballistic phase of the movement, should also be the first maximum. When the highest peak is *not* the first peak in the movement, the start point for the "trial" is moved temporally forward to begin at the minimum immediately

preceding the greatest maximum. Figure 3 shows a velocity profile for a movement where the start point of the "trial" is adjusted in this way.

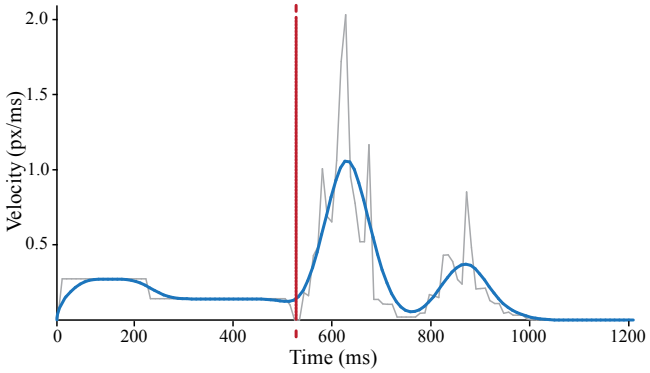

Figure 3. An aimed pointing movement whose start point is at the dashed line in accordance with Meyer *et al.*'s model [33].

The submovement maxima are smaller than the ballistic maximum and occur later in time (Figure 2, C-E). The number of submovements can be set in the Input Observer's configuration dialog. If the number of submovements is less than or equal to the maximum set by the researcher, the movement data is retained as a "trial." Movements that exceed the allowable number of submovements are discarded. The maximum allowed in our study was three, one extra than prescribed by Meyer *et al.*'s model [33]. Admittedly, uses of the Input Observer for alternative input devices (*e.g.*, eye-trackers) or users with motor challenges (*e.g.*, older users [45]) would need to adjust this parameter.

Identifying Targets

Pointing time (ms) can be calculated easily after extracting the velocity profile, but pointing error rate (%) and Fitts' throughput (bits/s) calculations require knowing target locations and dimensions. When a user clicks, the clicked target's coordinates can sometimes be obtained through the Windows Automation API. However, a number of common targets, such as buttons in web pages, are not accessible through the Automation API. Also, the Automation API can only provide information on targets that the user successfully acquired—it cannot discern whether a user may have missed in the first place.

To identify targets invisible to the Automation API and to identify pointing errors, we utilize Amazon's Mechanical Turk.[3] For each extracted "trial," a thumbnail screenshot 300 × 300 pixels in size is captured at the click point. (Hurst *et al.* [23] reported that 92% of on-screen targets are smaller than 300 × 300 pixels.) A dotted line representing the path of the mouse cursor up to the click-point and a picture of an arrow cursor are superimposed on the thumbnails (Figure 4). Even if the Automation API gives information about the widget the user clicked upon, the screenshots are sent to Mechanical Turk as the Automation results are not always reliable. For example, if the user misses the button she was aiming for and accidentally clicks in the empty space beside the button, the Automation API will return the coordinates and dimensions of the container or window in which the button resides.

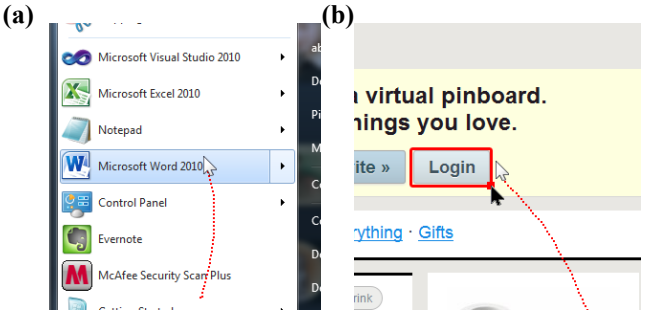

Figure 4. (a) Thumbnail showing the mouse trail, mouse cursor, and target. **(b)** Thumbnail showing a Turker (black cursor) dragging a bounding box over a web "Login" button, indicating a miss because the box fails to contain the white cursor.

Turkers communicate target locations, dimensions, and pointing errors in one swift step by dragging a bounding box around the target for which they think the user was aiming (Figure 4b). Turkers can also indicate with a checkbox that the intended target is not identifiable within the thumbnail image. In such cases, the "trial" is excluded from error rate and throughput calculations, regardless of whether the Automation API provided target dimensions.

Each image is sent to three Turkers to ensure the accuracy of responses. Targets' *left (x)*, *top (y)*, *width*, and *height* from each Turker for each "trial" are compared. When two or more sets of results have all four values within 10 pixels of each other, the results are considered "in agreement," and the mean values are calculated and kept. Mechanical Turk results are also compared to results from the Automation API. In the case of agreement between Mechanical Turk and the Automation API, the Automation results are used. If there is agreement among Mechanical Turk results but the Automation API returns *different* target dimensions, the Automation results are ignored—such results can be misleading for missed targets and targets to which the API has no access, such as buttons on web pages. In our study, the Automation API was incorrect for 33% of the targets identified correctly by Turkers.

Pointing errors are identified based on the target boundary results obtained. If the click-point is outside the identified target dimensions, the "trial" is marked as a pointing error. Of the 20,380 "trials" sent to Mechanical Turk by the Input Observer during our study, Turkers' bounding boxes agreed on target locations for 39.7% (8086) of the thumbnails. Those "trials" in the other 60.3% either had results failing to agree or Turkers agreed that no target could be identified. Based on a stratified random sample of 20 images from each of the 10 mousing participants (*cf.* Table 1), manual inspection showed that of the 12,296 thumbnails on which Turkers' bounding boxes did not agree, an estimated 94.5% contained targets that were genuinely ambiguous (*e.g.*, characters in a text editor).

[3] https://www.mturk.com/mturk/welcome

Creating *A*×*W* "Conditions" from Pointing Field Data

Once the Mechanical Turk results have been compiled and "trials" with inconclusive target boundaries have been discarded, remaining "trials" are grouped into "conditions" by target distance (A) and target size (W) to produce Fitts' law models for each user. To understand the need for $A \times W$ "conditions," we must review Fitts' law lab studies.

In lab studies, participants are presented with conditions defined by target distance (A) and target size (W), within which they perform numerous individual pointing trials [43]. Subsequently, when fitting Fitts' law to a participant, each nominal $A \times W$ condition provides one data point to the set of points on which Fitts' linear relationship between index of difficulty and movement time is established. The data point for a single $A \times W$ condition is plotted as $\left(ID_e, \overline{MT}\right)$, where $\overline{MT}$ is the average movement time of trials in the $A \times W$ condition and ID_e is the effective index of difficulty, which utilizes Crossman's [9] *post hoc* speed-accuracy normalization. ID_e is calculated as $\log_2(A_e / W_e + 1)$, where A_e is the average movement distance of trials in the given $A \times W$ condition and W_e reflects the spread of hits; in two dimensions, it is equal to $4.133 \times SD_{x,y}$, where $SD_{x,y}$ is the bivariate deviation of endpoints from their centroid [49].

The above procedure depends on having well-defined $A \times W$ conditions within which A_e and W_e can be calculated. These calculations are important for normalizing speed-accuracy tradeoffs and avoiding inflated throughputs that result from using nominal $ID = \log_2(A / W + 1)$ [31]. $A \times W$ conditions also enable us to retain error trials, rather than discarding errors, which must be done when using nominal ID. But for pointing "in the wild," there are no inherent $A \times W$ conditions, resulting in prior work only using nominal IDs and disregarding errors [7,22].

In seeking to utilize Crossman's [9] speed-accuracy correction and retain error trials, we enabled the Input Observer to cluster trials in our field data such that $A \times W$ "conditions" could naturally arise from the data itself. Given all pointing "trials" for a participant, the Input Observer clusters them into "conditions" using the nominal A and W parameters. (Prior work [35] recommends using $\sqrt{WH}$ for nominal W in two dimensions, so that is what we do.) After the $A \times W$ "conditions" are established, ID_e can be calculated as usual.

To find the groups of "trials" to serve as $A \times W$ "conditions" for a given user, that user's "trials," plotted as $(A, \sqrt{WH})$ ordered pairs, are clustered. The popular *k-means* algorithm [26] is not adequate because of the requirement that k, the number of clusters, be specified. Therefore, the *G-means* algorithm [19], which requires no specification of k, is used to cluster "trials." We augmented the G-means algorithm to use *k-means++* [2], which provides better initial cluster centers to G-means as G-means iteratively searches. G-means uses the Anderson-Darling test for normality [1], the statistic from which, called A^2, we adjust with Stephens' correction [44] for unknown means and variances, and D'Agostino's correction [10] for small samples in cases of $n \leq 25$. In using the Anderson-Darling test, G-means requires a significance value (α) to be specified, which we set based on the number of data points. For 400 or fewer data points, we use $\alpha = .10$. For more than 400 points, we use $\alpha = .05$. Corresponding critical values from D'Agostino [10] are 0.631 and 0.752, respectively. Therefore, non-normality is asserted if A^2, the outcome of the Anderson-Darling test, is greater than the critical value.

Within each cluster, outliers in $(A, \sqrt{WH})$-space are defined as being more than $1.5 \times SD_{x,y}$ from the centroid of the cluster. These outliers are removed. Similarly, temporal outlier "trials" with movement times longer than $1.5 \times SD_{MT}$ from the cluster's mean movement time are also removed. Twelve percent of all "trials" were identified as spatial or temporal outliers. To ensure that the "conditions" used to produce Fitts' law models still contain sufficient points after outlier removal, only clusters with 10+ surviving data points are retained. Fifty-seven percent of "trials" contained fewer than 10 data points after outlier removal.

"Conditions" where the ID_e is very small are also removed, as previous work has shown that Fitts' law is questionable in such cases [26]. We chose an ID_e of ≥ 1 as the threshold for inclusion in the throughput calculation. Eight percent of "trials" were removed because ID_e was less than 1.

In our study, the Input Observer formed, on average, 14.1 ($SD = 7.7$) clusters, or $A \times W$ "conditions," per participant. This number turned out to be close to the 18 conditions used in our lab study of 3 levels of A × 6 levels of W. On average, there were 12.7 "trials" in each cluster ($SD = 1.3$) and the ID_e ranged from 1.01 to 5.80.

Mouse Pointing and Privacy

At 300 × 300 pixels, the thumbnails extracted around click-points are large enough that a user's privacy could be compromised when the images are uploaded to Mechanical Turk. For example, when a user clicks in a mail program to read email, or clicks on a link in an online banking site, readable areas of potentially sensitive text may be visible.

We have taken several steps to protect users' privacy. First, Tessnet2,[4] a .NET wrapper for the Tesseract OCR library [40], is used to identify areas of text in each thumbnail. Those areas are then blacked out (Figure 5a). In addition, the Emgu wrapper[5] for OpenCV's [4] face detection library is used to find and black out faces (Figure 5b). Although the text and face detection processes go a long way towards protecting privacy, they are not perfect. Therefore, a narrow "filmstrip" showing thumbnails queued for Mechanical Turk remains docked at the right side of the desktop for users to observe while the Input Observer is running (Figure 5c). Users can see full-size images by hovering their mouse over thumbnails. Users can choose to remove

[4] http://www.pixel-technology.com/freeware/tessnet2/
[5] http://www.emgu.com/wiki/index.php/Main_Page

any thumbnail (and its associated "trial" data) by right-clicking on the thumbnail. Thumbnails are shown in the "filmstrip" for at least 2 minutes before uploading.

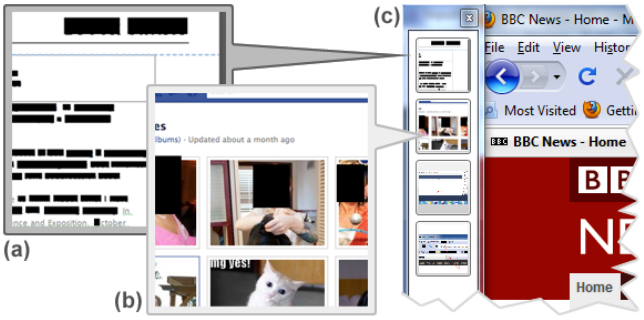

Figure 5. The Input Observer blacks out **(a)** text and **(b)** faces in thumbnails prepared for Mechanical Turk. **(c)** It also shows a "filmstrip" of thumbnails before uploading, enabling users to remove any thumbnails they wish.

INPUT OBSERVER EVALUATION

Our evaluation of the Input Observer comprised a weeklong field deployment and lab tests of participants' text entry and pointing performance. The lab results provided a baseline to which we compared the Input Observer data. It is important to note that human behavior differs between the lab and the field, and so there is no *a priori* reason to expect results from the Input Observer to exactly match those from the lab. Rather, our comparison is for gaining confidence that the Input Observer's measurements are not horribly awry.

Twelve participants ran the Input Observer on their own computers for the equivalent of one work-week. Five participants provided both text and mouse data, 5 provided only mouse data, and 2 provided only text data. All participants providing pointing data used an optical mouse. Table 1 summarizes participant demographics and the amount of field data collected.

P	Sex	Age	Observed	No. of Text "Trials"	No. of Mouse "Trials"	
					Recorded	Used
1	f	25	mouse	n/a	1436	254
2	m	23	both	48	3686	261
3	f	28	mouse	n/a	2595	265
4	f	34	mouse	n/a	935	67
5	f	29	both	68	608	47
6	m	53	mouse	n/a	743	34
7	f	22	both	108	3378	313
8	f	26	both	149	2609	243
9	f	34	both	145	762	273
10	f	24	mouse	n/a	1124	119
11	m	32	text	60	n/a	n/a
12	f	31	text	83	n/a	n/a
Mean		**30.3**		**94.43**	**1787.60**	**187.60**
SD		**8.7**		**40.60**	**1168.35**	**107.75**

Table 1. Participants and types and amounts of data collected.

Participants also completed lab tests of their text entry and pointing performance. The text entry data was collected using *TextTest* [46]. Each participant was presented with 55 phrases, including 5 practice phrases, from the MacKenzie and Soukoreff phrase set [30], which they were asked to type as quickly and accurately as possible. The text entry results were analyzed using *StreamAnalyzer* [46].

FittsStudy [49] was used for the pointing lab sessions. There were 3 levels of target distance (A = 256, 384, and 512 pixels) and 6 levels of target width (W = 8, 16, 32, 64, 96, and 128 pixels) resulting in 13 unique nominal *ID*s ranging from 1.59 to 6.02 bits. In each $A \times W$ condition, participants performed 23 trials, the first 3 of which were practice. Circular two-dimensional targets were used. The lab studies were conducted and analyzed based on prior work [43,49].

RESULTS

Text Entry Results

Table 2 shows text entry results per participant:

	WPM		Unc. Errors (%)		Cor. Errors (%)	
Participant	**Lab**	**Field**	**Lab**	**Field**	**Lab**	**Field**
2	97.22	101.26	0.00	0.07	2.39	0.22
5	80.02	92.54	0.14	0.08	0.84	0.44
7	72.46	80.60	0.06	0.93	3.90	0.55
8	73.24	70.62	0.34	0.18	2.79	1.14
9	79.52	84.64	0.15	0.30	3.53	2.24
11	54.81	57.73	0.17	0.00	3.44	0.92
12	63.70	76.76	0.00	0.38	0.80	0.81
Mean	**74.47**	**80.59**	**0.12**	**0.28**	**2.53**	**0.90**
SD	**13.39**	**14.28**	**0.12**	**0.32**	**1.27**	**0.67**

Table 2. Text entry results from *TextTest* and *StreamAnalyzer* [46], and field results from the Input Observer.

As Table 2 shows, the average text entry speed in the lab was 74.47 WPM (SD = 13.39). The Input Observer's average from the field was 80.59 WPM (SD = 14.28). The average uncorrected error rate in the lab tests was 0.12% (SD = 0.12). In the field it was 0.28% (SD = 0.32).

Although of lesser importance [47] (*see* p. 56), corrected errors were also measured. The average corrected error rate in the lab tests was 2.53% (SD = 1.27). In the field it was 0.90% (SD = 0.67). Recall that due to theoretical limitations, backspaces are the only error correction mechanism allowed in lab studies. In the field, however, error correction may also employ the mouse or cursor keys. Unfortunately, until theoretical breakthroughs incorporating such mechanisms are made, these error correction activities will remain elusive to tools like the Input Observer.

Mouse Pointing Results

As shown in Table 3 (next page), the average movement time in the lab was 971.34 ms (SD = 88.96). In the field, it was 870.07 ms (SD = 114.84). Note that movement times are dependent upon the A and W task parameters. A benefit of Fitts' law, of course, is that it is independent of A and W and only considers their ratio. That movement time is longer in the lab than the field indicates that lab targets may have been further away or smaller than those encountered in the field.

The average pointing error rate in the lab was 4.42% (SD = 3.56), which is near the ~4% rate prescribed for Fitts' law studies [27,43]. It was similar in the field, at 4.66% (SD = 1.76).

	Time (ms)		Error Rate (%)		Throughput	
Participant	**Lab**	**Field**	**Lab**	**Field**	**Lab**	**Field**
1	989.67	749.31	2.78	2.22	3.47	3.84
2	990.22	891.98	0.56	1.91	3.63	3.12
3	1024.11	779.17	1.39	5.51	3.32	3.60
4	948.56	1022.88	11.94	4.73	3.35	3.25
5	1097.61	889.19	1.11	7.64	3.19	3.12
6	1079.67	1099.43	4.17	3.24	3.03	2.94
7	986.44	800.09	2.50	5.85	3.45	3.80
8	926.22	854.66	7.78	5.44	3.49	3.70
9	821.83	861.54	5.83	4.61	3.99	3.47
10	849.11	752.41	6.11	5.42	3.87	3.63
Mean	**971.34**	**870.06**	**4.42**	**4.66**	**3.48**	**3.45**
SD	**88.96**	**114.84**	**3.56**	**1.76**	**0.29**	**0.32**

Table 3. Pointing results from *FittsStudy* [49] and field results from the Input Observer. Throughputs are measured in bits/s.

Recall that throughput is a combined speed-accuracy measure of pointing efficiency. The average throughput in the lab was 3.48 bits/s ($SD = 0.29$). In the field it was very similar, at 3.45 bits/s ($SD = 0.32$).

Table 4 gives the Fitts' law models for each participant's lab and field data. Figure 6 shows an example plot of MT (ms) by ID_e (bits) for one participant's extracted field data.

	Lab			Field		
Participant	***a***	***b***	R^2	***a***	***b***	R^2
1	27.75	253.50	0.95	275.29	157.49	0.78
2	27.63	267.82	0.97	210.21	224.65	0.71
3	153.69	224.36	0.95	381.54	137.04	0.77
4	-32.04	279.78	0.96	559.30	140.02	0.64
5	-61.57	335.11	0.95	548.70	124.12	0.66
6	106.55	294.37	0.98	899.23	59.72	0.51
7	-130.34	335.97	0.98	528.16	89.11	0.68
8	15.45	283.21	0.96	591.27	80.14	0.72
9	-13.60	255.30	0.97	394.59	154.42	0.90
10	5.78	258.92	0.94	450.39	122.72	0.95

Table 4. Fitts' law models from *FittsStudy* [49] and from the Input Observer.

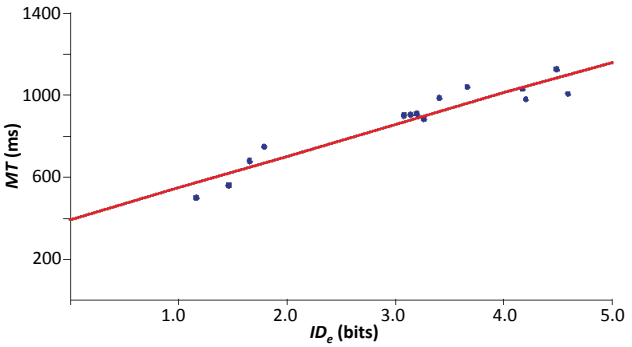

Figure 6. A plot of P9's extracted pointing data. Each blue dot represents one extracted $A \times W$ "condition." The red line represents the Fitts' law model.

DISCUSSION

Several measures produced by the Input Observer were very similar to the lab results, namely text entry uncorrected error rates, pointing error rates, and Fitts' throughputs. Also, the magnitudes of text entry speeds were within 8.42% on average across participants.

For text entry uncorrected error rates, both lab and field rates were nearly zero, but the field results were higher. Manual review of our log files showed that the differences were largely due to actual human behavior differences between lab and field, with more errors occurring in the field. One reason for this could be differences in the need for accurate spelling. In controlled studies, participants are told to be both fast and accurate. This resembles formal writing situations where accuracy is important. However, in informal writing, such as instant messaging, accuracy is less important and may even be undesirable, as error correction slows typed conversations.

We also manually crosschecked the Bing API to see how reliably it detected text entry errors. In most cases, the Bing API was good at catching errors. However, it occasionally marked error-free proper nouns, such as event names, as containing unknown errors. Although the Input Observer follows up on words not found by the Bing spell query with a web search to define the unknown word, Bing does not always return definition results for proper nouns. Google, on the other hand, does return definitions for proper nouns. Initially, the Input Observer used Google for text errors. However, Google changed its API during our project, necessitating the switch to Bing. Fortunately, proper nouns comprised only about 2% of words per participant.

Corrected error rates were different between the lab and field. Concern over corrected errors is mitigated by two points: (1) although corrected error rates give insight into the text entry process, they do not say much about the ultimate speed or accuracy of a method, as they are subsumed in WPM and, for methods with efficient error correction, do not imply that inaccurate text will be ultimately produced; and (2) forms of text entry error correction available in the field, such as using the mouse or text cursor keys, are not allowed in lab settings due to a theoretical inability to accommodate such behaviors in error rate analyses. Our manual review of field log files reveals that our participants frequently used the text cursor keys. Although we can see *that* a cursor key was used, we cannot see whether it was used to correct an error, make an edit, or even to scroll a web page or Adobe PDF file.

We are pleased with the Input Observer's ability to calculate pointing errors using crowdsourcing. Error calculation enabled the use of Crossman's [9] speed-accuracy normalization in calculating Fitts' throughputs. Although calculating pointing errors, Crossman's correction, and Fitts' throughputs required substantial infrastructure involving Automation APIs, crowdsourcing on Mechanical Turk, and data clustering with *G-means* [19] and *k-means++* [2], we were able to extract "trials" from field data and obtain performance measures that were similar to lab results. This was despite target layouts in the field being quite unlike the ISO 9241-9 circular layout of targets used in the lab [43] (*cf.* Figure 2, p. 754).

FUTURE WORK

A better understanding of application and environmental context and how it affects text entry and mouse pointing would be useful to this work. Supporting pointing devices other than conventional mice such as touchpads, trackballs, or isometric joysticks is also important, as these devices produce different submovement profiles [34]. An Input Observer for mobile devices would be useful for the study of "situational impairments" [39]. Similarly, enabling the Input Observer to segment text entry and mouse pointing for users with different abilities, such as older users [45], children [20], or people with motor impairments [24], is an important future step. The Input Observer already exposes parameters for the number of allowable submovements, and additional "knobs" are foreseeable. Finally, a challenging future topic is the extension of text entry error correction measurement to include not just the backspace key, but also the mouse and cursor keys.

CONCLUSION

The Input Observer is a potentially useful tool enabling field data to be gathered and analyzed unobtrusively and with sensitivity to privacy. We have shown that it is possible to extract lab-like "trials" and associated measures from everyday text entry and mouse pointing. Doing so entails inferring users' intentions with online resources such as web search and crowdsourcing. Our work may benefit researchers, device makers, and assistive technology specialists interested in evaluating and measuring performance during extended periods of "wild behavior."

ACKNOWLEDGEMENTS

This work was supported in part by the National Science Foundation under grant IIS-0952786. Any opinions, findings, conclusions or recommendations expressed in this work are those of the authors and do not necessarily reflect those of the National Science Foundation.

REFERENCES

[1] Anderson, T.W. and Darling, D.A. (1954). A test of goodness of fit. *J. American Statistical Association 49* (268), 765-769.

[2] Arthur, D. and Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. *Proc. SODA '07*. Philadelphia: Society for Industrial and Applied Mathematics, 1027-1035.

[3] Balota, D.A., Yap, M.J., Cortese, M.J., Hutchison, K.A., Kessler, B., Loftis, B., Neely, J.H., Nelson, D.L., Simpson, G.B. and Treiman, R. (2007). The English Lexicon Project. *Behavior Research Methods 39* (3), 445-459.

[4] Bradski, G.R. and Pisarevsky, V. (2000). Intel's Computer Vision Library: Applications in calibration, stereo, segmentation, tracking, gesture, face and object recognition. *Proc. CVPR '00*. Washington, D.C.: IEEE Computer Society, 796-797.

[5] Brown, B., Reeves, S. and Sherwood, S. (2011). Into the wild: Challenges and opportunities for field trial methods. *Proc. CHI '11*. New York: ACM Press, 1657-1666.

[6] Carter, S., Mankoff, J., Klemmer, S.R. and Matthews, T. (2008). Exiting the cleanroom: On ecological validity and ubiquitous computing. *Human-Computer Interaction 23* (1), 47-99.

[7] Chapuis, O., Blanch, R. and Beaudouin-Lafon, M. (2007). Fitts' law in the wild: A field study of aimed movements. *LRI Technical Report Number 1480*. Laboratorie de Recherche en Informatique. Orsay, France: Universite de Paris Sud.

[8] Choudhury, T., Borriello, G., Consolvo, S., Haehnel, D., Harrison, B., Hemingway, B., Hightower, J., Klasnja, P., Koscher, K., LaMarca, A., Landay, J.A., LeGrand, L., Lester, J., Rahimi, A., Rea, A. and Wyatt, D. (2008). The Mobile Sensing Platform: An embedded activity recognition system. *IEEE Pervasive Computing 7* (2), 32-41.

[9] Crossman, E.R.F.W. (1957). The speed and accuracy of simple hand movements. In *The Nature and Acquisition of Industrial Skills*, Crossman and Seymour (eds.). University of Birmingham: Final Report to the M.R.C./D.S.I.R. Joint Committee on Individual Efficiency in Industry.

[10] D'Agostino, R.B. (1986). Tests for the normal distribution. In *Goodness-of-Fit Techniques*, D'Agostino and Stephens (eds.). New York: Marcel Dekker, 367-420.

[11] Dixon, M. and Fogarty, J.A. (2010). Prefab: Implementing advanced behaviors using pixel-based reverse engineering of interface structure. *Proc. CHI '10*. New York: ACM Press, 1525-1534.

[12] Dumont, C., Vincent, C. and Mazer, B. (2002). Development of a standardized instrument to assess computer task performance. *American J. Occupational Therapy 56* (1), 60-68.

[13] Evans, A. and Wobbrock, J.O. (2011). Input Observer: Measuring text entry and pointing performance from naturalistic everyday computer use. *Extended Abstracts CHI '11*. New York: ACM Press, 1879-1884.

[14] Fisher, R., Perkins, S., Walker, A. and Wolfart, E. (2003). Gaussian smoothing. *Hypermedia Image Processing Reference*. Available at http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm

[15] Fitts, P.M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *J. Experimental Psychology 47* (6), 381-391.

[16] Fitts, P.M. and Radford, B.K. (1966). Information capacity of discrete motor responses under different cognitive sets. *J. Experimental Psychology 71* (4), 475-482.

[17] Froehlich, J., Chen, M.Y., Consolvo, S., Harrison, B. and Landay, J.A. (2007). MyExperience: A system for *in situ* tracing and capturing of user feedback on mobile phones. *Proc. MobiSYS '07*. New York: ACM Press, 57-70.

[18] Gajos, K.Z., Weld, D.S. and Wobbrock, J.O. (2010). Automatically generating personalized user interfaces with SUPPLE. *Artificial Intelligence 174* (12-13), 910-950.

[19] Hamerly, G. and Elkan, C. (2004). Learning the *k* in *k*-means. In *Advances in Neural Information Processing Systems 16*, Thrun, Saul and Schölkopf (eds.). Cambridge, MA: The M.I.T. Press, 281-288.

[20] Hourcade, J.P. (2006). Learning from preschool children's pointing sub-movements. *Proc. IDC '06*. New York: ACM Press, 65-72.

[21] Hurst, A., Hudson, S.E. and Mankoff, J. (2007). Dynamic detection of novice vs. skilled use without a task model. *Proc. CHI '07*. New York: ACM Press, 271-280.

[22] Hurst, A., Mankoff, J. and Hudson, S.E. (2008). Understanding pointing problems in real world computing environments. *Proc. ASSETS '08*. New York: ACM Press, 43-50.

[23] Hurst, A., Hudson, S.E. and Mankoff, J. (2010). Automatically identifying targets users interact with during real world tasks. *Proc. IUI '10*. New York: ACM Press, 11-20.

[24] Hwang, F., Keates, S., Langdon, P. and Clarkson, P.J. (2004). Mouse movements of motion-impaired users: A submovement analysis. *Proc. ASSETS '04*. New York: ACM Press, 102-109.

[25] Jansen, A., Findlater, L. and Wobbrock, J.O. (2011). From the lab to the world: Lessons from extending a pointing technique for real-world use. *Extended Abstracts CHI '11*. New York: ACM Press, 1867-1872.

[26] Lloyd, S.P. (1982). Least squares quantization in PCM. *IEEE Trans. Information Theory 28* (2), 129-137.

[27] MacKenzie, I.S. (1992). Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction 7* (1), 91-139.

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

[29] 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.

[30] 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.

[31] MacKenzie, I.S. and Soukoreff, R.W. (2003). Card, English, and Burr (1978)—25 years later. *Extended Abstracts CHI '03*. New York: ACM Press, 760-761.

[32] MacKenzie, I.S. and Isokoski, P. (2008). Fitts' throughput and the speed-accuracy tradeoff. *Proc. CHI '08*. New York: ACM Press, 1633-1636.

[33] Meyer, D.E., Abrams, R.A., Kornblum, S., Wright, C.E. and Smith, J.E.K. (1988). Optimality in human motor performance: Ideal control of rapid aimed movements. *Psychological Review 95* (3), 340-370.

[34] Mithal, A.K. and Douglas, S.A. (1996). Differences in movement microstructure of the mouse and the finger-controlled isometric joystick. *Proc. CHI '96*. New York: ACM Press, 300-307.

[35] Murata, A. (1996). Empirical evaluation of performance models of pointing accuracy and speed with a PC mouse. *Int'l J. Human-Computer Interaction 8* (4), 457-469.

[36] Palen, L. and Salzman, M. (2002). Voice-mail diary studies for naturalistic data capture under mobile conditions. *Proc. CSCW '02*. New York: ACM Press, 87-95.

[37] Rogers, Y. (2011). Interaction design gone wild: Striving for wild theory. *interactions 18* (4), 58-62.

[38] Rosenbaum, D.A. (1991). Aiming. In *Human Motor Control*. San Diego, CA: Academic Press, 205-216.

[39] Sears, A., Lin, M., Jacko, J. and Xiao, Y. (2003). When computers fade... Pervasive computing and situationally-induced impairments and disabilities. *Proc. HCI Int'l '03*. Mahwah, NJ: Lawrence Erlbaum, 1298-1302.

[40] Smith, R. (2007). An overview of the Tesseract OCR engine. *Proc. ICDAR '07*. Washington, D.C.: IEEE Computer Society, 629-633.

[41] Soukoreff, R.W. and MacKenzie, I.S. (2001). Measuring errors in text entry tasks: An application of the Levenshtein string distance statistic. *Extended Abstracts CHI '01*. New York: ACM Press, 319-320.

[42] 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. *Proc. CHI '03*. New York: ACM Press, 113-120.

[43] Soukoreff, R.W. and MacKenzie, I.S. (2004). Towards a standard for pointing device evaluation, perspectives on 27 years of Fitts' law research in HCI. *Int'l J. Human-Computer Studies 61* (6), 751-789.

[44] Stephens, M.A. (1972). EDF statistics for goodness of fit and some comparisons. *J. American Statistical Association 69* (347), 730-737.

[45] Walker, N., Philbin, D.A. and Fisk, A.D. (1997). Age-related differences in movement control: Adjusting submovement structure to optimize performance. *J. Gerontology: Psychological Sciences 52B* (1), 40-52.

[46] Wobbrock, J.O. and Myers, B.A. (2006). Analyzing the input stream for character-level errors in unconstrained text entry evaluations. *ACM Trans. Computer-Human Interaction 13* (4), 458-489.

[47] Wobbrock, J.O. (2007). Measures of text entry performance. In *Text Entry Systems*, MacKenzie and Tanaka-Ishii (eds.). San Francisco: Morgan Kaufmann, 47-74.

[48] Wobbrock, J.O., Kane, S.K., Gajos, K.Z., Harada, S. and Froehlich, J. (2011). Ability-based design: Concept, principles, and examples. *ACM Trans. Accessible Computing 3* (3), 9:1-27.

[49] Wobbrock, J.O., Shinohara, K. and Jansen, A. (2011). The effects of task dimensionality, endpoint deviation, throughput calculation, and experiment design on pointing measures and models. *Proc. CHI '11*. New York: ACM Press, 1639-1648.

[50] Zhai, S. (2004). Characterizing computer input with Fitts' law parameters—the information and non-information aspects of pointing. *Int'l J. Human-Computer Studies 61* (6), 791-809.

[51] Zhai, S., Kong, J. and Ren, X. (2004). Speed-accuracy tradeoff in Fitts' law tasks—on the equivalency of actual and nominal pointing precision. *Int'l J. Human-Computer Studies 61* (6), 823-856.