Fitts’ Law in the Wild: A Field Study of Aimed Movements
Olivier Chapuis, Renaud Blanch, Michel Beaudouin-Lafon

To cite this version:
Olivier Chapuis, Renaud Blanch, Michel Beaudouin-Lafon. Fitts’ Law in the Wild: A Field Study of Aimed Movements. 2007. <hal-00612026>

HAL Id: hal-00612026
https://hal.archives-ouvertes.fr/hal-00612026
Submitted on 28 Jul 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
ABSTRACT

This paper presents the first field study of aimed movements in graphical user interfaces, designed to get better insight into pointing in the real (electronic) world and assess the validity of Fitts’ law in the wild. We unobtrusively collected kinematic data from 24 users over several months, and segmented it into a table of 2 million movements. We show that Fitts’ law is indeed robust for modeling pointing performance, provided that an adequate noise reduction process is applied. We introduce the length-distance index (LDI) to take into account the fact that many movements are not straight, and we introduce an extension of Fitts’ law that includes an LDI term. We also show evidence of the effect of cognitive tasks in pause and click time, and the sensitivity of differences in performance, e.g., across input devices, to LDI. Altogether, these findings provide a firm ground to better assess the validity of results obtained in the laboratory.

ACM Classification Keywords
H. Information Systems H.5 Information Interfaces and Presentation H.5.2 User Interfaces (H.1.2, I.3.6)

Author Keywords
Fitts’ law, field study, pointing performance, UI logs

INTRODUCTION AND RELATED WORK

Target acquisition using a pointing device is one of the fundamental tasks in graphical user interfaces (GUIs). Since Card et al.’s seminal work [4], a large body of research has been dedicated to improving pointing performance through new devices and/or interaction techniques. At the heart of this research program lies Fitts’ law [2]. Fitts’ pointing paradigm reduces target acquisition to a unidimensional task characterized by two independent variables: target width (W) and target distance (D). Fitts’ law predicts that the movement time (MT) to acquire a target is a linear function of the task’s index of difficulty (ID):

\[ MT = a + b \times ID \]

where \( ID = \log_2 \left( \frac{D}{W} + 1 \right) \). (1)

The index of difficulty fully characterizes the task: higher values of ID, i.e. longer distances and/or smaller targets, require more time and are therefore “harder”. The intercept (a) and the slope (b), on the other hand, depend on the context of execution of the task, i.e. the user, the input device, etc.

Fitts’ law is the most successful quantitative law used in HCI [21,26] and has proved extremely robust across a wide range of conditions. It has been extended, among other things, to 2D pointing [22,10,11], 3D pointing [9] and multiscale navigation [12]. It is used to compare input devices, e.g., [23], pointing techniques, e.g., [3] and types of users, e.g., [13]. It is even the basis for a standardized experiment used to define the throughput of input devices by the International Organization for Standardization (ISO) [18].

Fitts’ pointing paradigm focuses exclusively on the motor control aspect of pointing. Users are instructed to execute aimed movements as fast as possible, with a single target prominently displayed. As a result, Fitts’ law models pointing in a GUI in the ideal situation where the user knows exactly where the target is and points directly at it. This is hardly a typical situation: pointing “in the wild” involves other factors than simply the size and position of the target, such as deciding what is the target, dealing with interference from the environment or planning for higher-level tasks. Since so much of the HCI literature depends on the results of controlled experiments based on Fitts’ pointing paradigm, it is legitimate to ask if we can expect the results of such experiments to transfer to actual use in everyday GUIs.

The purpose of the work reported in this paper is to test the ecological validity of Fitts’ law and more generally to study and analyze aimed movements in the field as opposed to in the laboratory. To our knowledge, it is the first field study of pointing based on a long term collection of pointing data. While field studies are common in HCI and tools exist to collect user data, e.g., for usability studies [19], we are not aware of any field study of the motor aspects of GUIs. For example, Hurst et al. [14] log mouse data to distinguish between novice and skilled users, but their goal is to dynamically adapt the software. Hutchings et al. [15] log window management activity to study the use of multiple monitors, but focus on higher-level tasks.
To conduct the study presented in this paper, we have logged the activity of 24 users over several months with two software probes and analyzed the collected data, which represents over 2 million aimed movements and about 1 billion pixels covered by the cursor (about 219 miles or 352 km at 72 dots per inch). The remainder of this paper describes the methodology we have used to collect and segment the data. Then we proceed with the analysis itself. We introduce the length-distance index (LDI) to take into account the fact that many movements are not straight. We show that Fitts’ law is indeed robust for modeling pointing performance. We then find evidence that the cognitive part of pointing tasks affect performance and correlate this degradation with LDI. This leads to a generalization of Fitts’ law that includes an LDI term. Finally, we compare performance across input devices and widget types and conclude with some future work.

SYSTEM INSTRUMENTATION
We have developed two software probes targeting two different desktop environments: the X Window System under Linux (X Window) and Apple’s Mac OS X (OS X). The probes are unobtrusive, i.e., they run in the background and do not require modifying any of the applications run by the user. The information collected is a kinematic log similar to that of VibeLog [15]: the successive on-screen positions of the cursor with their timestamp, the state changes of the pointing device buttons, and the position, size and stacking order of the windows on screen. We also log the geometry of the widget under the cursor at the time of each click, however for X Window this information is not always available due to technical limitations.

X Window Probe
On X Window we used wmtrace, a freely available tool developed to monitor window management activities [5]. wmtrace logs pointer events by monitoring the X Window protocol and window information by monitoring the window management protocols: the latter include window decorations (title bar, window buttons, window borders), the buttons in the taskbar and in the pagers. At the beginning of our study (early 2005), it was only possible to log the geometry and type of widget under the cursor for GTK applications.

Mac OS X Probe
On OS X, we have developed a probe that tracks the pointing device events using Apple’s IOKit framework to access the low-level USB-HID drivers. The windows’ geometry is collected when a button’s state changes using the accessibility API provided by the ApplicationServices/HIService framework. Additional information such as stacking order which is unfortunately not available through the accessibility API is collected using an undocumented functionality of the ApplicationServices/CoreGraphics framework.

The geometry and type of the widget under the cursor is also collected using the accessibility API when a button is pressed. Sixty-nine widget “roles” (Button, MenuBar, MenuItem, DockItem etc.) and eight “subroles” (MinimizeButton, ToolbarButton, etc.) are defined by the API, allowing us to study subsets of the collected data, e.g., all the clicks on the close button of a window. The collected data is sufficient to reconstruct the complete cursor trajectory and the state of the screen each time a button is pressed (Figure 2).

MOVEMENT SEGMENTATION
In order to study aimed movements in the real world, we have to extract such movements from the kinematic data we have collected, i.e. to segment the data into candidate movements and select those that correspond to a pointing task. This step is critical since the quality of the segmentation will determine the validity of our future analyses. In particular, we must be careful and avoid using hypotheses that underlie our analyses, e.g., that aimed movements follow Fitts’ law, to segment the source data.

An aimed movement begins when the user decides to reach a target. This cognitive process does not leave an explicit trace in the data we collect, so we need heuristics to identify the beginning of a movement. The end of an aimed movement in GUIs almost always corresponds to an explicit...
selection action, i.e. a button press (or a button release in the case of a drag). Some movements may not have such an explicit marker, e.g., when hovering over an item to see its tooltip. In the present study, we have decided to focus on aimed movements that end with a button press (↓). We exclude drags because they often correspond to non-aimed movements such as scrolling or inking, and we exclude hovering because we do not have a good heuristic to distinguish it from pauses. The following sections describe in more detail the segmentation process.

**Beginning of Movement**

To our knowledge, the literature does not address the identification of the beginning of an aimed movement: experimental protocols make the beginning of the movement explicit, e.g., as the first movement after the display of the target. In our case, given a button press at time \( t_1 \), we know that the movement does not start before the button was last released \( t_2 \). The movement may have started after \( t_1 \) however, for example if the user has moved the cursor to help with reading or in case of external interruptions. Therefore we need to analyze the cursor trajectory between \( t_1 \) and \( t_2 \).

Points where the cursor velocity is zero, called pauses\(^2\) are of particular interest to segment the trajectory. Defining the beginning of the movement as the last pause in the trajectory does not work for two reasons. First, aimed movements consist of one or several submovements\(^{[24]}\), which may be separated by (short) pauses. So a pause could be a submovement transition. Second, we have noticed in our data as well as in direct observations that users make long pauses shortly before \( t_1 \), including immediately before \( t_1 \), as if they were mentally confirming their action before clicking.

This leads us to use two parameters for the segmentation: the stop time and the verification distance. Let us call \( P \) the position of the cursor at \( t_1 \) and \( p \) the position of the cursor at a pause. We define a stop as a pause longer than stop time that occurs at a distance \( d(p,P) \) greater than the verification distance. The beginning of the movement is then the last stop before \( t_1 \). Figure 3 illustrates a cursor trajectory with time running horizontally and the distance along the trajectory running vertically. There is one stop, \( P_1 \), one pause in the middle of the movement, \( P_2 \), and two pauses close to \( t_1 \), \( P_3 \) and \( P_4 \). Since \( P_1 \) is the last stop before \( t_1 \), the beginning of the movement is the end of that pause (\( t_b \)). We call movement time (MT) the time between \( t_b \) and \( t_1 \), verification time (VT) the total duration of the pauses that occur within the verification distance, and click time (CT) the duration of the pause, if any, that occurs immediately before \( t_1 \). Finally, we call pause time (PT) the total duration of the pauses that occur during the movement excluding those within the verification distance, i.e., in this example, \( P_2 \).

Given these definitions, we need to pick adequate values for the two parameters of the segmentation, stop time and verification distance. Hwang et al.\(^{[17]}\) have studied the mouse movements of motion-impaired users with able-bodied users\(^2\) A pause is a period of 50 ms or more with no cursor motion. This delay is needed to account for the irregular sampling of the cursor.

![Figure 3. Segmentation of a cursor trajectory](image)

as control. This study gives some statistics about “pauses” between submovements for able-bodied users. Although no precise definition of a pause is given, the mean number of pauses in a movement was about 1 and the mean time of a pause was about 70 ms. Using the kinematic data collected for a previous Fitts experiment, we determined that with a verification distance of 10 pixels, the mean pause time (PT) was 75 ms and 95% of the pauses were less than or equal to 240 ms. The mean verification time (VT) was 185 ms, with 592 ms for the 95% quantile.

For the study presented in this paper, we chose a verification distance of 10 pixels and a stop time of 300 ms. As the stop time seemed quite long, we used our replay tool to make sure that it was a reasonable value through extensive observation of the data. Note that the verification distance means that we missed movements whose amplitude is less than 10 pixels. We believe this is acceptable since such small movements are of limited interest. Also, since we have no restriction on VT, we may have extremely long movements where the click time largely dominates the movement time.

**End of Movement**

As stated earlier, we only analyze movements that end with a button press, making the segmentation of the movement’s end trivial. In order to distinguish among different types of final actions at the end of the movement, we measure the duration (\( \text{d}_{\text{drag}} \)) and distance traversed (\( \text{d}_{\text{drag}} \)) while the button is pressed, i.e. between \( t_1 \) and the following \( t_2 \). Using an analysis of the distributions of \( \text{d}_{\text{drag}} \) and \( \text{d}_{\text{drag}} \), we determined thresholds to distinguish among the following final actions: a drag (\( \text{d}_{\text{drag}} > 5 \) pixels) or a click (\( \text{d}_{\text{drag}} \leq 5 \) pixels). Clicks are further subdivided into two categories: regular clicks (\( \text{d}_{\text{drag}} \leq 300 \) ms), and long clicks (\( \text{d}_{\text{drag}} > 300 \) ms). Finally, if the difference between \( t_1 \) and the following \( t_1 \) is less than or equal to 250 ms, that movement is discarded and the previous final action is concatenated with the current one as a combination, such as a double click.

Since we plan to conduct a Fitts’ analysis of the aimed movements that have been segmented, we need to know the geometry of the aimed target. We call targeted movements those movements for which we have reliable information about the target, i.e. movements for which we can be fairly confident
that the widget under the cursor was the actual target. Our data logs capture information about the widget under the cursor. We use the widget type to classify the movement as a targeted movement. For example, if the widget is a text area, the actual target is likely to be a character in the text as opposed to the whole text area. Since we do not have information about the actual target in this case, we do not record the movement as being targeted and eligible for a Fitts’ analysis.

**DATA SUMMARY**

We now present the data we have collected and a first analysis of the results of the segmentation. We developed our own software for the segmentation and thoroughly verified the results. Analyses were conducted with the R[^2] and JMP[^3] software packages. After segmentation, our main data table contains over 2 million rows (one per segmented movement).

**Users Characteristics**

Twenty-four users participated in the study, 20 men and 4 women, all right-handed. All users were either students, colleagues or engineers from different institutions and were expert computer users. Ten participants used their computers in two configurations, one in three configurations. A configuration is a combination of computer, desktop environment (for Linux users), pointing device (mouse or trackpad) and number of screens. For example, a user may switch to another environment under Linux; a laptop user may use a second monitor and a mouse when working in his office. This resulted in 36 different configurations, listed on the left of Table 1. 9 participants and 13 configurations under X Window, 15 participants and 23 configurations under OS X.

Participants used the probing software from several weeks to several months, and we regularly collected the log files. The second part of Table 1 summarizes the results of the segmentation to give an idea of the magnitude of the data (the total duration of logged data is not included because it would include long periods of time where users are away from their computer): total number of movements (mvts), number of movements with long values of \( \text{LD} \) (with \( \text{LD} > 100 \)), and percentage of all movements. The last section of the table lists the same data for targeted movements, i.e. movements for which we know the target. These represent 22% of all movements. The last column (mean Fitts equation) will be explained later.

**Analysis of Final Actions**

Each movement ends with a final action such as a click, a double click or a long click. Most final actions (96.89 ± 3.39%) are carried out with the left button[^5]. The variability across configurations stems from the differences among desktop environments. On OS X, most participants use a single-button trackpad or mouse, or have not configured the right and middle buttons. Only one OS X user uses the right button more than 1% of all clicks (3.07%). A few OS X users use the middle button, with a maximum of 4.7% for a user of Exposé, a window management function used to navigate among windows. On X Window, the right button is used in 4.43 ± 2.41% of all clicks and the middle button in only 1.76 ± 1.56% of all clicks. The middle button pastes the primary selection and all of the X Window users know this feature, with a maximum of 4.64% of all clicks.

A large majority of left button actions are single clicks, then double clicks and then long clicks. On OS X we have 77.06 ± 8.02% clicks, 15.54 ± 6.55% drags, 4.49 ± 2.53% double clicks, 1.42 ± 1.29% long clicks and less than 2% other combinations, e.g., triple clicks. On X Window we have 72.56 ± 6.04% clicks, 17.75 ± 4.66% drags, 4.71 ± 1.84% double clicks. 2.2 ± 1.29% long clicks and 2.75 ± 1.7% other combinations.

**The Length Distance Ratio and Length Distance Index**

One of the consequences of pointing in 2D is that the movement trajectory is not fully determined by the specification of the target as in a traditional 1D Fitts’ experiment. While a straight line may not be the fastest way to reach a target because of the motor control involved in pointing, it is a good approximation. To measure the deviation from this ideal trajectory, we use the length-distance ratio (LD), computed as the ratio of length of the movement to direct distance between the beginning and end points of the movement.

We observed that LD could be huge. We have 701 movements with \( LD > 100 \) and 1.42% of the targeted movements with \( LD > 10 \). These large values are mostly due to movements with short distances: all but 8 movements with \( LD > 100 \) have a distance \( D < 80 \), and 75% of the movements with \( LD > 10 \) have a distance \( D \leq 100 \). Obviously, it is easier to obtain a large \( LD \) ratio when \( D \) is small.

Analysis of movements with large values of \( LD \) using the wmttrace-view tool revealed patterns such as large circular, spiral or zig-zag movements. These could correspond to a user showing something on the screen, or to an attempt at locating the cursor after having lost it (particularly on a large screen or 2-monitor configuration) by moving it and then proceeding with a standard aimed movement.

![Figure 4. LD (left) and LDI (right) distributions (D ≥ 100).](http://www.jmp.com)

Table 1 lists the mean values of LD per configuration. Figure 4(left) shows the distribution of LD across all aimed

[^1]: [http://www.R-project.org](http://www.R-project.org)
[^3]: [http://www.R-project.org](http://www.R-project.org)
[^5]: We use the notation \( m ± σ \) where \( m \) is the mean and \( σ \) the standard deviation of the value across configurations. Unless otherwise noted, we first take the mean for each configuration and then compute the mean and deviation across configurations.
movements with $D \geq 100$. It is highly skewed and looks like a power law. Looking at the quantiles, we find that the mean (across configurations and $D \geq 100$) of the medians is $1.15 \pm 0.06$. The mean of the first quartile is $1.05 \pm 0.02$ and the mean of the third quartile is $1.48 \pm 0.18$. The difference between OS X and Window is minor and we obtain similar values when considering only targeted movements. Thus, while a large fraction of the movements are almost straight, a significant number are not. For example, 10% of movements with $D \geq 100$ have $LD \geq 2.46$. We compared this data to that from a previous controlled Fits experiment and found the mean value of $LD$ to be $1.045 \pm 0.09$ (median $= 1.025$) with $90\%$ of the movements having $LD \leq 1.178$.

The $LD$ distribution looks like a power law, has a very long tail and strongly depends on distance. It is thus difficult to interpret. Instead, we propose to consider the length distance index (LDI) defined as follows:

$$LDI = (\frac{L}{D} - 1)^2$$

This transform of $LD$ breaks the power law, normalizes the small $LD$ values and concentrates large $LD$ values. Figure 5 (right) shows the resulting plot. The mean $LDI$ is $0.66 \pm 0.06$ and the median, close to the mean, is $0.61 \pm 0.07$, matching the peak of the distribution. As we will see later, this index will prove useful in our analyses.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Aimed movements</th>
<th>Targeted movements</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>enschler et al.</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1 X (delle)</td>
<td>mouse</td>
</tr>
<tr>
<td></td>
<td>1.3954</td>
<td>428.821</td>
</tr>
<tr>
<td></td>
<td>1.28</td>
<td>4223</td>
</tr>
<tr>
<td></td>
<td>1.28</td>
<td>297116</td>
</tr>
<tr>
<td>2</td>
<td>4 X (nome)</td>
<td>mouse</td>
</tr>
<tr>
<td></td>
<td>1.28</td>
<td>13708</td>
</tr>
<tr>
<td></td>
<td>1.31</td>
<td>1513</td>
</tr>
<tr>
<td></td>
<td>1.51</td>
<td>119356</td>
</tr>
<tr>
<td>3</td>
<td>1.67</td>
<td>7235749</td>
</tr>
<tr>
<td></td>
<td>1.55</td>
<td>11305</td>
</tr>
<tr>
<td></td>
<td>1.56</td>
<td>8973</td>
</tr>
<tr>
<td>4</td>
<td>1.16</td>
<td>294703</td>
</tr>
<tr>
<td></td>
<td>1.28</td>
<td>51159</td>
</tr>
<tr>
<td></td>
<td>1.31</td>
<td>59636</td>
</tr>
<tr>
<td>5</td>
<td>1.48</td>
<td>225162</td>
</tr>
<tr>
<td></td>
<td>1.48</td>
<td>3589</td>
</tr>
<tr>
<td></td>
<td>1.51</td>
<td>190987</td>
</tr>
<tr>
<td>6</td>
<td>1.42</td>
<td>123972</td>
</tr>
<tr>
<td></td>
<td>1.42</td>
<td>52756643</td>
</tr>
<tr>
<td>7</td>
<td>1.28</td>
<td>71588</td>
</tr>
<tr>
<td></td>
<td>1.32</td>
<td>28036</td>
</tr>
<tr>
<td>8</td>
<td>1.42</td>
<td>19808</td>
</tr>
<tr>
<td></td>
<td>1.39</td>
<td>8740</td>
</tr>
<tr>
<td>9</td>
<td>1.51</td>
<td>23001</td>
</tr>
<tr>
<td></td>
<td>1.42</td>
<td>12411</td>
</tr>
<tr>
<td>10</td>
<td>1.28</td>
<td>6440</td>
</tr>
<tr>
<td></td>
<td>1.35</td>
<td>80136</td>
</tr>
<tr>
<td>11</td>
<td>1.28</td>
<td>1613</td>
</tr>
<tr>
<td></td>
<td>1.58</td>
<td>885</td>
</tr>
<tr>
<td>12</td>
<td>1.44</td>
<td>3589</td>
</tr>
<tr>
<td></td>
<td>1.56</td>
<td>2074</td>
</tr>
<tr>
<td>13</td>
<td>1.41</td>
<td>18173</td>
</tr>
<tr>
<td></td>
<td>1.41</td>
<td>19420</td>
</tr>
<tr>
<td>14</td>
<td>1.44</td>
<td>12499</td>
</tr>
<tr>
<td></td>
<td>1.54</td>
<td>1436967</td>
</tr>
<tr>
<td>15</td>
<td>1.66</td>
<td>66031</td>
</tr>
<tr>
<td></td>
<td>1.23</td>
<td>40029</td>
</tr>
<tr>
<td>16</td>
<td>1.57</td>
<td>53782</td>
</tr>
<tr>
<td></td>
<td>1.69</td>
<td>22431</td>
</tr>
<tr>
<td>17</td>
<td>1.42</td>
<td>45124</td>
</tr>
<tr>
<td></td>
<td>1.33</td>
<td>20866</td>
</tr>
<tr>
<td>18</td>
<td>1.59</td>
<td>15692</td>
</tr>
<tr>
<td></td>
<td>1.30</td>
<td>9696</td>
</tr>
<tr>
<td>19</td>
<td>1.58</td>
<td>58091</td>
</tr>
<tr>
<td></td>
<td>1.38</td>
<td>13646</td>
</tr>
<tr>
<td>20</td>
<td>1.59</td>
<td>2141</td>
</tr>
<tr>
<td></td>
<td>1.28</td>
<td>9341</td>
</tr>
<tr>
<td>21</td>
<td>1.60</td>
<td>11094</td>
</tr>
<tr>
<td></td>
<td>1.39</td>
<td>7036</td>
</tr>
<tr>
<td>22</td>
<td>1.66</td>
<td>22255</td>
</tr>
<tr>
<td></td>
<td>1.33</td>
<td>8290</td>
</tr>
<tr>
<td>23</td>
<td>1.59</td>
<td>10464</td>
</tr>
<tr>
<td></td>
<td>1.66</td>
<td>3390</td>
</tr>
<tr>
<td>24</td>
<td>1.44</td>
<td>10447</td>
</tr>
<tr>
<td></td>
<td>1.31</td>
<td>37161</td>
</tr>
</tbody>
</table>

Table 1. Summary of participants, configurations and collected data. Aimed movements are all the movements resulting from the segmentation. Targeted movements are those for which we know the geometry and type of the target.

![Figure 5. Distribution of movement direction at the beginning of the movement (top) and from end to end (bottom).](image-url)

Variability in Movement Direction

Figure 5 plots two distributions of movement directions: the direction at the beginning of the movement (top) and the overall direction of the movement, i.e. the direction of the straight line going through the beginning and end points of the movement (bottom). While the aliasing due to the discretisation of cursor positions may explain the privileged directions at the beginning of the movement (0, 90, 180, 270°), we do not explain the privileged directions (horizontal and vertical) for the full movement.
We find similar distributions when considering only targeted movements and/or when removing movements with small distances. However, when analyzing movements by widget type, we find that some widget types elicit movements with horizontal, vertical and sometimes diagonal directions because of their position on the screen (e.g., OS X Menu Bar items or TaskBar buttons), the geometry of the widget (e.g., large and thin TitleBar vs. small TitleBarButton) or their position relative to the cursor (e.g., popup menu items).

**Pointing Task Variables**

By studying targeted movements only, i.e. those movements for which we know the geometry of the target, we can study the characteristics of the underlying pointing task, i.e. the independent and dependent variables of Fitts’ law, with the notable exception of error rate, since by definition all of our movements are hits.

Since we are studying a 2D pointing task, we must decide what distance, width and ID to consider. There are a number of generalizations of Fitts’ law to 2D in the literature, including MacKenzie & Buxton [22], Acott & Zhai’s bivariate pointing [11] and Grossman & Balakrishnan’s probabilistic model [10]. While the latter was not applicable to our data since it requires hits and misses we tried several of the other formulas and found little differences among them. Therefore we opted for the simplest model, MacKenzie & Buxton’s minimal size: the size of the target is \( W_{\text{min}} \), the smallest dimension of the target, and the index of difficulty is \( ID_{\text{min}} \), computed from Equation 1 with \( W_{\text{min}} \) instead of \( W \).

Figure 6 shows the distribution of distance, \( W_{\text{min}} \) and \( ID_{\text{min}} \) for the targeted movements of our data set. Note the two peaks around \( ID = 1 \) and \( ID = 5 \). Most of our targets have a \( W_{\text{min}} \) of about 20 pixels, corresponding to buttons, title bars, scrollbars, etc. There are probably more targeted movements for which we know the geometry of the target, we can study the characteristics of the underlying pointing task, i.e. the independent and dependent variables of Fitts’ law , with the notable exception of error rate, since by definition all of our movements are hits.

Figure 6 also shows the distribution of click time (\( CT \)), verification time (\( VT \)) and movement time (\( MT \)). The distributions of \( CT \) and \( VT \) are very similar. Indeed, \( CT \) contribute 80% of \( VT \). More importantly, we note that \( CT \) is quite large: after removing outliers by considering movements with a movement time of 5 sec at most, the mean click time \( CT \) is 277 ± 78 ms and 20% of these targeted movements have \( CT \geq 380 \) ms. This was confirmed by examining some of the data with \texttt{wintrace-viewer}. We will come back to this observation later in the paper.

**Fitting Fitts’ Law**

In this section, we investigate the role of \( ID \) on dependent variables. We especially check its relationship to movement time (\( MT \)), i.e. whether Fitts’ law holds for our data. The data set is the set of targeted movements, for which we have target information.
Table 2. Fitts’ parameters for subsets of the data (mean and gev 100-percentile process). For the last three lines, conditions are cumulative.
finding an item, e.g., in a menu or in the taskbar; (iii) double checking before a dangerous action, e.g., when closing a window or deleting a file. Such cognitive activities may take place during the movement [4] and are likely to interfere with motor control and therefore pointing performance. While we do not have direct traces of this cognitive activity, we can search for clues in our data.

Observation of movement ends shows that users often spend a fairly long time before clicking once they have stopped in the target. Analysis of variance shows that CT and VT depend on target size and, more surprisingly, on distance (CT and VT decrease when W and D increase). However, these dependencies seem to be weak; we cannot find, and do not think there is any, robust law that can predict CT and VT based on the geometric properties of a pointing task. Since we do not have a motor-control explanation for these long click and verification times, and since they are not observed in traditional Fitts’ experiments, we strongly suspect that they are due, at least in part, to cognitive activity during pointing.

Another measure that may indicate cognitive activity during the movement is pause time (PT). We found a very strong correlation between PT and LD, i.e., pause time is longer when movements are not straight. Using the length-distance index LD instead of LD, we found a very good linear correlation between PT and LD: $PT = 4.753 + 318.2 \cdot LD$, $r^2 = 0.990$ (10-percentile mean fit for all the data) and $PT = -4.667 \pm 5.19 + 309.9 \pm 82.6 \cdot LD$, $r^2 = 0.919 \pm 0.053$ (10-percentile mean fit by configuration). We verified on subsets of our logs that pauses often correspond to changes in direction of movement, which could explain this correlation. These changes are most likely due to cognitive activity, e.g., changing one’s mind while reaching for a target.

Since LD correlates with longer pauses and therefore longer movement time, we are interested in combining it with ID to create a combined index of difficulty (CID) that accounts both for the difficulty of the motor task and the “distraction” caused by cognitive activity. Using an ANOVA for the model $MT \sim ID + LD + ID \cdot LD$ with the configuration as random factor, we found, as expected, a strong effect of ID ($F_{1,465275} = 106852, p < 0.001$), an even stronger effect of LD ($F_{1,465276} = 125626, p < 0.001$) and an ID × LD interaction effect ($F_{1,465261} = 16328, p < 0.001$). The interaction between ID and LD can be easily explained in terms of Fitts’ Law: as LD increases the slope and intercept of the Fitts equation increase as well. This is particularly visible if we use the measure $MT - VT$: Figure 10 shows the successive fit lines when we restrict the data to successive LD intervals (8-percentile).

Analysing the results of the ANOVA leads to the following definition of CID:

$$CID = ID + 8 \cdot LD + 4 \cdot ID \cdot LD$$

which gives the following model: $MT = 362 + 49 \cdot CID$, $r^2 = 0.302$ for the complete data set (vs. $r^2 = 0.117$ when fitting with ID). Since we have shown that CT and VT in-

The last 3 lines of Table 2 show what happens when gradually constraining the data set according to movement properties. The first constraint ($LD < 1.2$) selects movements that are almost straight. This substantially reduces the intercept. The second constraint (# pause < 3) further selects movements with no submovements. This reduces the slope. The third constraint ($CT < 300$) further selects movements with short click times. Together, these constraints amount to selecting those movements that are closest to those observed in a controlled experiment where movements are straight, direct and without pause at the end. The slope and intercept then match the typical values reported in the literature for controlled experiments. Overall these analyses show that Fitts’ law indeed holds in the wild.

Fitts’ Law and the Cognitive Part of a Pointing Task
Using a reduction process, we have seen in the previous sections that Fitts’ law can adequately model motor control and therefore pointing performance. While we do not have direct traces of this cognitive activity, we can search for clues in our data.

Observation of movement ends shows that users often spend a fairly long time before clicking once they have stopped in the target. Analysis of variance shows that CT and VT depend on target size and, more surprisingly, on distance (CT and VT decrease when W and D increase). However, these dependencies seem to be weak; we cannot find, and do not think there is any, robust law that can predict CT and VT based on the geometric properties of a pointing task. Since we do not have a motor-control explanation for these long click and verification times, and since they are not observed in traditional Fitts’ experiments, we strongly suspect that they are due, at least in part, to cognitive activity during pointing.
introduce noise that we cannot yet control, we obtain even better fits with $MT-CT$ ($r^2 = 0.497$ vs. 0.207 with $ID$) and $MT-VT$ ($r^2 = 0.541$ vs. 0.231 with $ID$). We obtain similar improvements when restricting the fit to individual users, input device, and configuration. Since straight movements, which are the norm in controlled experiments, have an $LDI$ close to zero, these results suggest that $CIL$ is a possible generalization of Fitts’ $ID$.

COMPARATIVE PERFORMANCE ANALYSIS

We now turn to a comparative analysis of pointing performance. First we analyze differences among configurations by looking at the effect of desktop environment, input device, and configuration. Since straight movements, which are the norm in controlled experiments, have an $LDI$ close to zero, these results suggest that $CIL$ is a possible generalization of Fitts’ $ID$.

Desktop Environment, input device and Screens

We already noted in our Fitts’ analysis a significant difference in performance between mouse and touchpad in favor of the mouse. In order to analyze the effects of both the desktop environment and input device, we performed a full factorial ANOVA of movement time with $ID$, $LDI$, $Input$ and $Env$ as factors and $User$ as a random factor (some users have used both a mouse and touchpad).

We found no significant effect of $Env$ on movement time. This may be seen as a proof of the common wisdom that desktop environments have become very similar, as well as a testament to the robustness of our logging tools.

We found strong simple effects of $ID$, $LDI$ and $Input$. A post-hoc test reveals a difference in mean of 369 ms between the mouse and touchpad. However, we found an interaction between $Input$ and $Env$. For the touchpad, the mean movement time is 533 ms for X Window and 205 ms for OS X but there is no significant difference for the mouse. However, this difference vanishes if we remove the click time ($CT$) and use $MT-CT$ as measure: there is no difference between the X Window user and all the mouse users. On the other hand, we still see a significant difference between the touchpad and the mouse for OS X users. We must be careful however in interpreting these results since there is only one X Window touchpad user.

We further investigate interactions between $ID$, $LDI$ and $Input$ by restricting our analysis to OS X users. We find $ID \times Input$ and $LDI \times Input$ interaction effects. In both cases, as the index ($ID$ or $LDI$) increases, the difference between mouse and touchpad becomes smaller. If we take $MT-CT$ as a measure, we find strong simple effects of $ID$, $LDI$ and $Input$ and a difference in means of 105 ms. We find no interaction between $ID$ and $Input$ but an interaction between $LDI$ and $Input$ that can be explained by the reduction of the difference between mouse and touchpad for large $LDI$.

It is not surprising that, as $LDI$ grows, the difference between mouse and touchpad shrinks: as movements are less straight and less efficient, the inherent capabilities of the input device have less impact on performance. It is more surprising however that the same thing happens with $ID$. This is consistent with Yun & Lee [27] but in contradiction with Epps [7] and Amer et al. [2]. A possible interpretation is that OS X has an optimized transfer function for the touchpad. Note also that 50% of the difference is contained in the click time showing the importance of click techniques for touchpads [23].

We found no robust effect when investigating the effect of screen configuration even when considering single-screen movements vs. cross-screen movements. It is clear however that in a dual screen configuration, there is a primary screen where almost all clicks are performed.

Widget Type

We now turn our attention to the effect of widget type. The results presented here are based on mouse movements only, however they are still valid when considering the complete data set, or touchpad users only, or each user separately. We used $ID$, $LDI$ and $Widget$ as factors and the configuration as a random factor and performed several ANOVAs with different subsets of widget types.

First, we focus on widgets with different shapes: the window title bar (long rectangle) and buttons (square), the window border (42 ms) and the resizing grip (small triangle at the bottom right of the window). We test two definitions of Fitts’ index difficulty for 2D pointing: $ID_{\min}$ [22], which uses the smallest dimension, and $ID_{2,1,1}$ [1], which takes into account the aspect ratio.

With $ID = ID_{\min}$, we find significant simple effects of $ID$, $LDI$ and $Widget$ and small $Widget \times ID$ and $Widget \times LDI$ interaction effects caused by the window border becoming faster than the other widgets as $ID$ or $LDI$ increases. A post-hoc Tukey HSD test shows the following ordering (> means faster with the difference indicated in parentheses): border (42 $\pm$ 13 ms) > title bar (74 $\pm$ 13 ms) > resizing grip (175 $\pm$ 13 ms) > window button. This shows a substantial difference in mean of 248 $\pm$ 13 ms between the window title bar and the window buttons.

$$7ID_{2,1,1} = \log_2(\sqrt{(D/W)^2 + (D/H)^2} + 1)$$
With \( ID = ID_{2,1,1} \), we find similar simple and interaction effects, but no significant difference in mean between border and resizing grip and between resizing grip and title bar (we still have a small difference between title bar and border: \( 49 \pm 12 \) ms). There is still a large difference between the title bar and buttons (\( 169 \pm 5 \) ms for a grand mean of \( 1500 \) ms). This difference does not depend on \( ID \) but decreases as \( LDI \) grows. We believe this difference can only be caused by cognitive activity. Indeed, closing or iconifying a window has more dire consequences than moving or resizing it, in that the cost of undoing the action is higher. This explanation is supported by the fact that the difference is mostly concentrated in the click time \( CT \) (\( 135 \pm 8 \) ms).

We then test the particular case of widgets that are on the border of the screen, such as the menu bar or the dock icons on OS X: since the cursor is blocked by the border of the screen, these widgets effectively have a semi-infinite size on OS X: since the cursor is blocked by the border of the screen, such as the menu bar or the dock icons \( [25] \). Another special type of widget are menu items. First, popup menus appear next to the cursor and have low \( ID \)s, which should improve performance. Second, long menu items often require visual scanning, which should impair performance. We found evidence of these two hypotheses: menu items are one of the fastest widgets for small \( ID \) and one of the slowest for large \( ID \), i.e. we observe a strong interaction of \( ID \) by \textit{Widget} when considering the menu item widget type against the other types. This supports designs such as the split menu \( [25] \).

**CONCLUSION**

We have described what we believe is the first field study of aimed movements. Using two unobtrusive software probes, we have collected kinematic logs from 24 users in 36 hardware/software configurations over several months. We have described our strategy to segment this data into a collection of over 2 million aimed movements and a subset of about half a million targeted movements, i.e. movements for which we know the geometry and type of the target. The goal of this study was to gain insight into real-life pointing and assess the validity of the results of controlled experiments of pointing techniques in the real world. In particular, we wanted to test the validity of Fitts’ law in the wild.

The results of our study can be summarized in the following findings. First, we introduced the length-distance index \( (LDI) \) as a complementary tool to the index of difficulty \( (ID) \) in order to better understand aimed movements that are not straight. Second, we showed that Fitts’ law was a good predictor of mean performance even when using separate quantiles and \textit{gev} locations, demonstrating the remarkable robustness of the law. This strongly supports the fact that results gathered in the lab with controlled Fitts’ experiments can be valid in the field. However, we also measured large intercepts of Fitts’ regression lines and found a lot of noise in the data. We used the \( LDI \) to obtain better fits, first by filtering out movements with large \( LDI \), then by introducing a variant of Fitts’ law that includes an \( LDI \) term. Since movements are mostly straight in Fitts’ studies \( (LDI = 0) \), this new law arguably generalizes Fitts’ law. Finally, we found unexpectedly high values for pause time \( (PT) \) and click time \( (CT) \), which we attribute to the cognitive load associated with the task at hand. We found a linear correlation between pause time \( (PT) \) and the length-distance index \( (LDI) \) and we often observed that the differences in pointing performance decreased as \( LDI \) increased. We also found that \( CT \) was strongly dependent on the nature of the target.

While this work is mainly theoretical, it has immediate practical applications. For example, many movements with high \( LDI \) correspond to users “shaking the mouse” to find the cursor, typically after a long pause, calling for more efficient ways of locating the cursor, especially for large screens and multiple-display settings. Reducing click time is also an interesting challenge as it does not seem to be linked to pure motor control. We intend to pursue this work by analysing other aspects of the data, by testing the combined index of difficulty in an experimental setting and by using our findings to guide the design of new interaction techniques.

**REFERENCES**


