UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

PREDICTION OF USER INTENTIONS WHEN OPERATING WITH THE CURSOR

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

LUCAS CEZARD
Norman, Oklahoma
2017

Downloaded from SHAREOK. Persistent Handle URL: https://hdl.handle.net/11244/52771

PREDICTION OF USER INTENTIONS WHEN OPERATING WITH THE CURSOR

A THESIS APPROVED FOR THE
SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING

BY

_____
Dr. Ziho Kang, Chair

_____
Dr. Theodore B. Trafalis

_____
Dr. Charles D. Nicholson

© Copyright by LUCAS CEZARD 2017
All Rights Reserved.

Acknowledgements

First, I would like to thank Dr. Ziho Kang for his mentoring and support during this research. Thank you for the opportunity you gave me to pursue the research in a field I am passionate about. I would also acknowledge Dr. Charles D. Nicholson and Dr. Theodore B. Trafalis for agreeing to be part of my committee.

I would also like to thank all the students who gave of their time to participate in the experiment that makes this work complete.

Finally, I want to thank my parents who made this life-changing experience possible.

Table of Contents

Acknowledgements iv

Table of Contents v

List of Tables viii

List of Figures ix

Abstract xi

Chapter 1: Introduction and literature review 1

1- Background 1

1.1 Pattern identification and target prediction 1

1.2 Prediction of movement time 3

1.3 Usability testing 4

2- Other related researches 5

2.1 The role of aging in cursor manipulation 5

2.2 Gaze and cursor alignment 7

2.2 Eye-tracking applications 9

Chapter 2: Motivation 10

1- Real time access to mouse-tracking data 10

2- Auto-completion tools 11

3- Research questions 12

Chapter 3: Methodology 13

1- Create a mouse-tracker 13

1.1 Generating an interface 13

1.2 Technical details 14

1.3 Data interpretation 15
2- Define and detect patterns and cues 17
2.1 Cursor position 17
2.2 Checkmark cue 18
2.3 Bracket-shaped cue 20
2.4 Angle of the movement 21
2.5 Circular shape 22
3- Applications 23
3.1 Scrolling a document 23
3.1.1- Position-based solution 24
3.1.2- Cue-based solution 25
3.2 Selecting text 26
3.2.1- Bracket shaped cue 28
3.2.2- Circular cue 28
3.2.3- Checkmark cue 29
3.3 Navigate on a web browser 30
3.3.1- Opening a Google menu tab 31
3.3.2- Complete recurring movements 33
3.4 Final deliverable 35
Chapter 4: Experimentation 37
1- Recruitment 37
2- Technical details 37
3- Navigation improvement tool 38

4- Text selection tool 39
5- Scrolling tool 39
6- Usability questionnaire 39
Chapter 5: Results 41
1- Familiarity with the touchpad 41
2- Perceived difficulty of the tasks 41
3- Intuitiveness of the patterns 42
4- Auto-completion tools usability scores 43
5- Influence of the tools on the computer use 44
6- Comparison of the completion times 45
6.1 Movement times 45
6.2 Text selection task 47
6.3 Scrolling task 47
7- Comparison of the completion rates 48
Chapter 6: Discussions 50
Chapter 7: Future work 54
Conclusion 56
References 58
Appendix A: Usability questionnaire 61

List of Tables

Table 1. The different questions of the usability questionnaire classified by the usability measure they evaluate..... 40

Table 2. Theoretical movement time resulting from Fitts' law. 46

Table 3. Summary of the completion rates for each task for the experimental group.... 48

List of Figures

Figure 1. X and Y coordinates of the cursor for three different readers..... 5

Figure 2. Interfaces for the mouse-tracker..... 15

Figure 3. X or Y position of the cursor against time for three participants..... 16

Figure 4. Detection of the presence of the cursor around a given point (X, Y). 18

Figure 5. Top and bottom part of the screen as described earlier..... 19

Figure 6. Checkmark cues used for the auto-scrolling tool..... 20

Figure 7. Bracket-shaped cue. 20

Figure 8. Illustration of the use of the movement's angle to detect the target. 22

Figure 9. The different angles computed to detect a circular cue..... 23

Figure 10. Successive steps to automatically scroll the page with the cue detection..... 27

Figure 11. An example of how the bracket-shaped cue needs to be drawn around text. 28

Figure 12. Left: an example of how the checkmark cue needs to be drawn around text. Right: explanation on where the cursor is moved for the beginning of the text selection. 29

Figure 13. The interface created for the navigation improvement tools. 30

Figure 14. Steps for the automatic opening of Google News..... 31

Figure 15. The homepage divided into four quadrants and the movements that can be identified and autocompleted (colored by target point)..... 34

Figure 16. Interface of the final auto-completion tool..... 36

Figure 17. Histogram presenting the score given for tasks difficulty. 42

Figure 18. Histogram presenting the score given for patterns intuitiveness. 43

Figure 19. Boxplots of the usability scores given for each auto-completion tool..... 43

Figure 20. Histogram presenting the score given for the influence of the tools on the navigation. 44
Figure 21. Mean and standard error plots for each movement completion time, by group, in seconds.. 46
Figure 22. Mean and standard error plots for text selection task completion time, by group, in seconds. 47
Figure 23. Mean and standard error plots for scrolling task completion time, by group, in seconds. 48

Abstract

Studies have shown that when we operate the cursor on a computer, several factors such as the type of device used (i.e. mouse or touchpad), aging (young vs. old) or motor-impairment can hinder performances. More precisely, using the touchpad can be difficult for any user, even for the most basic tasks, due to the absence of scrolling wheel and the reduced amplitude of cursor movement.

To cope with these issues, I developed a set of tools to increase the usability of the touchpad by analyzing mouse-tracking data. More specifically, several movement patterns or cues were predefined and when they were detected, they would trigger the auto-completion of the related task which includes navigating on a web browser, selecting text and scrolling.

The usability experiment conducted to assess the ease-of-use of the created tools and to compare the performances of participants showed promising results. Participants appreciated the help of the auto-completion tools and when they were able to trigger these tools, they were significantly faster. In particular, when moving the cursor to the URL address bar they even outperformed Fitts' law predictions. However, it appeared that participants needed several attempts to draw certain cues correctly hence a longer completion time.

Keywords: human factors, usability, mouse-tracking, cues, pattern detection, auto-completion.

Chapter 1: Introduction and literature review

Mouse-tracking is useful for many purposes such as usability testing and human cognition studies. Through the recording of the cursor trajectory and other features like clicks and scrolling, we can analyze how we interact with computers or gain insights on the hesitations we face while choosing between several options. Mouse-tracking is also widely used to improve the design of menus, applications, and web pages. Indeed, it is an interesting technique for usability testing as it provides meaningful information (cursor fixations, duration of a movement, speed of the movement) that can be easily computed and interpreted. Moreover, it provides similar results to eye tracking with cheaper and easier-to-use devices.

The following sections summarize the literature related to this study and the opportunities for future work. The first section dwells on the use of mouse-tracking to determine users' behavior, prediction of cursor's movement time and usability testing that are the primary focus of this research. Although it is not part of this study, it is necessary to consider the influence of aging on computer use, the link between eye and cursor position and the applications of eye-tracking.

1- Background

1.1 Pattern identification and target prediction

Numerous studies focused on the user's interest during web search task to offer a more personalized experience or improve the quality of the results. Mueller et al. (2001) used mostly the back-and-forth movements of the cursor and they considered that these movements were representative of the hesitations of the user between several results.

Then, they used the insights from these hesitations to determine what interested the most the user and provide personalized content. Huang et al. (2011) used the same principle and focused on the use of hovering data to define the user's interest and improve the quality and the relevance of the search results. Additionally, Rodden et al. (2008) were able to identify three cursor movement patterns when examining SERP that are also linked to eyes movements. These three patterns consist of following the eye horizontally or vertically and marking cursor a promising result with the cursor while the eyes continue exploring the results. Finally, Guo et al. (2008) focused on the determination of the query intent. But instead of using only mouse clicks, they used cursor movements as they discovered that cursor movements had specific characteristics depending on the query intent. For instance, the vertical range of cursor movements was larger in the case of informational tasks as the user was likely to explore all the results provided in the SERP. Then, they used their findings to create a classifier that outperforms previous models using clicks and related features.

Pusara et al. (2004) demonstrated how powerful, cursor movement patterns can be. Indeed, they developed, using machine-learning techniques, a model that was able to identify a user only thanks to his or her use of the mouse. The features included cursor position, speed and angle of movement, clicks and scroll wheel use. With this model, they were able to re-authenticate computer users with their mouse habits instead of requiring that they re-enter their passwords.

It is important to note that identifying movement patterns often relies on visualization of eye-tracking data as recommended by Räihä et al. (2005). Static visualization enable the plotting of the coordinates of the gaze superimposed on the

content displayed on the screen. This gives valuable insights on the specific points of interest that the user considered. But these visualizations can also be used to plot mouse-tracking data. Figure 1 shows how different cursor movement patterns can be identified for a reading task. More precisely, one used the cursor as a reading aid (green plus signs), one left the cursor inactive (blue circles), and one moved the cursor only line-by-line (red crosses).

In another context, studies focused on the possibility of creating a model to predict in real time the target of the cursor movement. Murata (1998) used the trajectory and more precisely the angle of the cursor movement to determine potential targets and move automatically the cursor over the most likely target. The results showed that this model decreased the pointing time but highlighted the difficulty to make accurate predictions when targets were close from each other. Moreover, the trade-off between pointing time and accuracy was discussed as the results of the experiment proved that increasing the number of samples used to predict the trajectory of the cursor did not always increased the accuracy of the said predictions. Ziebart et al. (2012) explained how they developed a new probabilistic model using, machine learning techniques, to predict the target of the movements with the help of more features. This model was interesting as it overcame the difficulty of predicting accurately the target when several small targets were close to each other.

1.2 Prediction of movement time

Another important field of interest is the modeling of the time required to complete a certain task with the mouse. In particular, the time needed to move the cursor

from a starting point to a precise target can be modeled by Fitts' law and numerous studies proved that this law is well-fitted to predict the movement's duration. It is also widely used to compare the performances of a new prediction model to the previous theoretical time needed to complete the movement. MacKenzie (1992) reviewed the state of the art regarding Fitts' law and the promising refinements to have a more accurate law to predict movement time. In this study, the author also demonstrated how Fitts' law can help for usability experiment as three different techniques to delete a file on a Mac computer were compared using Fitts' law. These ideas are also found in the study of Boritz et al. (1991) where the authors included the angle of the target as a mitigating effect on the movement time before validating their model in a usability experiment of a pie-shaped menu. They proved that this new model was accurate to predict the movement time and that the angle between the cursor and the target effectively played a key-role in the time required to reach the target. However, some studies highlighted the accuracy issues of Fitts' law for small targets (Oel et al., 2001). To cope with this issue, a new model has been developed to infer the movement time from a power law including the distance from the target and some parameters that depend on the width of the target.

1.3 Usability testing

Finally, to assess the quality of the tools presented in this paper, it is necessary to conduct a usability testing. The metrics used to quantify usability are often debated but three main characteristics emerged in the ISO/IEC 9126-4 Metrics norm as effectiveness, efficiency, and satisfaction. Effectiveness refers to how well the user performed the specified task, efficiency is the efforts required to achieve the given effectiveness and

satisfaction quantifies the ease of use. However, each of these constituent elements can be measured in various ways; for example, there are dozens of standardized questionnaires to evaluate satisfaction (Sauro et al., 2005). Moreover, some usability measures were introduced for a specific field of research (Bevan, 2006). More precisely, for our target users, some usability metrics were presented to account for the relationship between the elderly and IT and included a measure of trustworthiness and non-intrusiveness in the usability evaluation (Holzinger et al., 2008). To reduce the dimensions of usability data, Sauro et al. (2005) discussed how some common metrics can be standardized and, using PCA analysis, how we can use only one measure as a combination of all usability metrics weighed with the first principal components coefficients while retaining up to 60% of the original variance.

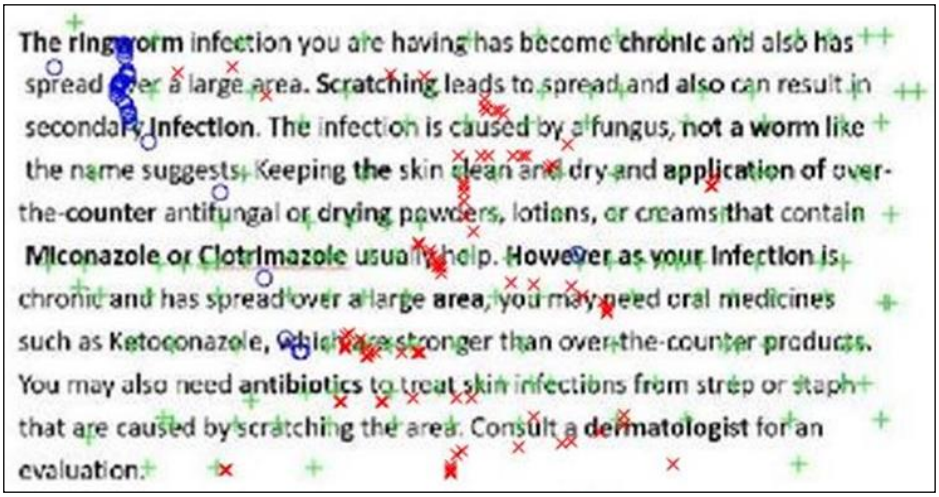


Figure 1. X and Y coordinates of the cursor for three different readers.

2- Other related researches

2.1 The role of aging in cursor manipulation

Several studies shed light on the influence of aging on the motor capabilities and its consequences on cursor control. Armbrüster et al. (2007) tested two tasks, pointing

and clicking and pointing-dragging-dropping with two different devices including a touchpad for different age categories. Their results showed that for touchpad users, middle-aged participants (40 to 65 years old) were 45% slower for the first task and 39% slower for the second task compared to younger participants (20 to 32 years old). Another study completed the results from the previous study by experimenting various tasks such as pointing, clicking, double-clicking and dragging (Smith et al., 1999). Once again, aging was linked to a loss of motor control that triggered significantly slower use of the mouse. In this study, it was explained by the fact that older participants tended to cover a larger distance to reach a target, in particular, because they did more sub-movements and they also had a lower accuracy in pointing at targets. From these conclusions, it became clear that modeling the user's behavior to be able to predict his or her next target and anticipate the cursor movements is crucial. Indeed, if we can detect a pattern in the user behavior, we can later identify this pattern and complete the task automatically.

Other researches were conducted to adapt the cursor behavior to the challenges faced by older or motor-impaired users. Trewin et al. (2006) developed a tool that aims at reducing clicking errors like slipping the mouse before releasing a button, clicking while moving or pushing the wrong button. To tackle these issues, their feature prevents clicking if the cursor is moving over a certain speed and "freezes" the cursor in its position when a target is clicked. The results from the pointing and clicking experiment performed by motor-impaired users demonstrated that the tool developed improved target selection and worked as an error filter for overlapping clicks and clicks performed while the cursor is moving. Gajos et al. (2008) went even further and offered personalized interfaces based either on preferences or abilities. They tested the usability of these personalized interfaces

compared to a baseline interface for pointing and clicking tasks with able-bodied and motor-impaired participants. The results showed that even able-bodied participants performed better with personalized interfaces and in particular with the abilities based one. Motor-impaired participants also reduced their error rate when using adapted interfaces and found it easier and less tiring to use interfaces that were designed specifically to suit their capabilities. However, each of these solutions has some limitations. Indeed, the tool developed by Trewin et al. (2006) only addresses clicking problems and the adaptive interfaces proposed by Gajos et al. (2008) can only replace a set of interfaces and requires a long setting phase.

2.2 Gaze and cursor alignment

To confirm the link between eyes and cursor movements, several studies focused specifically on the gaze/cursor alignment. First of all, Quetard et al. (2016) studied the processes involved during visual search to find a specific target in an everyday life scene. Findings showed that eyes and cursor movements can be linked mostly in the case of noisy pictures or when the target was placed in an unusual location. In these cases, they observed that more eye fixations were made to verify the presence of the target and in parallel, the cursor movements were slower and deviated more from a straight trajectory, indicating hesitations in the decision-making process.

But most of the researches regarding gaze/cursor alignment concentrated on SERP (Search Engine Results Page), how we examine these results and webpages. Cooke (2006) asked participants to find several information on the Washington State Department of Licensing web site and determined that when the cursor appeared on screen, 69% of

the time, it matched the gaze position. Similarly, Chen et al. (2001), recorded gaze and cursor positions of participants browsing four websites with different designs. They defined seven regions related to the different points of interest on a webpage and found that 84% of the regions visited by the cursor were also visited by the eyes, suggesting a strong correlation between eyes and cursor movements. To a larger extent, Huang et al. (2012) distinguished several cursor movement patterns when browsing a SERP and compared gaze/cursor alignment for each behavior. In particular, the gaze and cursor positions were the closest when the cursor was used to click or perform specific action like scrolling with a median distance around 75 pixels. On the contrary, when the cursor was inactive, the gaze was at a median distance of 233 pixels from the cursor.

The strong correlation between eyes and cursor movements lead to multiple research to predict the gaze position. The first model was proposed by Guo et al. (2010) and consisted of a classifier indicating if gaze position is aligned or not with the cursor for a sample point. Different methods were used and trained with data collected on navigational and informational tasks including the cursor position but also velocities along each axis. The results showed that the LogitBoost algorithm outperformed all other methods and reached an accuracy of 77% for the most precise model (gaze and cursor were considered aligned if the distance was less than 100 pixels). Huang et al. (2012) used their findings on the influence of the cursor behavior on the gaze/cursor alignment to create a model that includes this feature. However, they reached a surprising conclusion since it appeared that it was easier to predict the X coordinate of the gaze than the Y coordinate although every study found that the gaze/cursor alignment is stronger along the Y-axis.

2.2 Eye-tracking applications

Extensive research was conducted on eye-tracking and its applications, and they are important as eye and mouse-tracking are closely related. Moreover, it showed the importance of these technologies in usability testing or to improve learning techniques for instance. In particular, Holmqvist et al. (2005) and Vitu et al. (1995) used eye-tracking to explore the factors that influence reading behaviors in the case of newspapers. More specifically, they studied the influence of the content's complexity, the general design of the document and some more specific factors such as the text size and color or the positioning of the images and figures on the reader's attention. It is asserted that reading patterns and the quality of the reading were not fully dependent on the content but depended more on the general layout and size of the text and the presence of images along the text. Other studies like Kang et al. (2014) focused on the analysis of air traffic controllers scanning patterns of their radar screen. Finally, Wilson et al. (2016) also analyzed data from radar screen scanning patterns but in the scope of meteorology and more precisely to understand how meteorologists determined if a meteorological phenomenon was dangerous and how they triggered adequate alert messages.

Chapter 2: Motivation

Being able to understand how we use the mouse is crucial since it is the most common way to interact and control computers. Therefore, every computer user needs to be comfortable when using a mouse or touchpad to be able to control his or her computer. Moreover, most of the tasks involving the mouse are repetitive; for instance, while navigating on a web browser we are likely to enter several URL addresses to access different websites and move the cursor to the URL address bar. Thus, we can argue that there is a set of cursor movements that are often performed and characteristic of common tasks in computer use.

In the following sections, we will discuss the objectives and requirements of the different tools developed in this research before detailing the research questions.

1- Real time access to mouse-tracking data

Mouse-tracking is a mature technology, easy to understand and easy to implement. Jon Freeman, professor at NYU, has developed a mouse-tracking software used worldwide by more than 3,000 researchers[1]. However, this software only allows for post hoc treatment and analysis of the mouse-tracking data. In the context of pattern identification, this is an issue as it does not enable the detection of cues when they occur. Moreover, some computations like the velocity of the cursor along each axis are not demanding and do not hinder the performances of the software. Thus, for this research, it is necessary to create a mouse-tracking software with a real time access to the trajectory to compute the necessary features for pattern recognition.

[1] http://www.mousetracker.org/

This tool was also designed in the scope of the research taking place at the Human Factors and Simulation laboratory at the University of Oklahoma[2]. To serve future researches, this mouse-tracker will potentially be coupled with an eye-tracker for real time analysis of both eyes and cursor movements. Consequently, the mouse-tracking tool was developed using Matlab R2016 so that it will be compatible with the eye-tracking software.

2- Auto-completion tools

In this work, I explored the difficulties that older or motor-impaired people can face when using a laptop and its touchpad to point or click a small icon. In particular, I used mouse-tracking to create tools to spare this population demanding actions with the mouse. I identified three major areas of improvement. First I developed an application that facilitates the scrolling task. This task seemed hard to perform in the case of laptops used without an additional mouse and thus deprived of scrolling wheel. In this case, the user needs to click on a relatively small icon on the scrolling bar which requires a precise movement. The second idea I worked on is the assistance for text selection. Indeed whether it is performed with the touchpad or a traditional mouse, we need to maintain the left button pressed while we move the cursor toward the end of the text we want to select. The simultaneous actions required can be challenging for people encountering motor impairment. To tackle this issue, I developed a tool that automatically maintains the said left button pressed while the user is moving the cursor to select some text. The last idea I explored is the auto-completion of predefined movements. I assumed that we can identify

[2] https://humanfactors.oucreate.com/home.html?r

a set of movements that are demanding or repeated regularly when using a web browser. Then, when the pattern of movements is detected, the movements can be automatically performed.

3- Research questions

The development of these tools and the subsequent usability testing helped us answer the following questions: is it possible to create a mouse-tracker and use its data in real time? Is it possible to facilitate the scrolling task by avoiding the scrollbar? Is it possible to facilitate text selection? Can we identify some repetitive and difficult tasks to perform when using the touchpad and develop an automatic completion of such tasks? What is the gain obtained with these tools?

Chapter 3: Methodology

This chapter details the different steps followed to conduct this study. First, I developed a mouse-tracking device, using Matlab, which meets the specific requirements for pattern recognition. Then the focus is set on the different cues and patterns that are used in the auto-completion tools. Finally, details are provided on the tasks that are considered for auto-completion and which patterns or cues are used to trigger these tools.

1- Create a mouse-tracker

This first sub-section provides insight on the creation of the mouse-tracker, how we can generate an interface with Matlab, the technical challenges to determine the cursor position and time but also how mouse-tracking data can be interpreted.

1.1 Generating an interface

Matlab has a dedicated tool, GUIDE that allows the creation of Graphical User Interfaces (GUI). Each element such as textboxes or buttons can be placed on a blank canvas using drag-and-drop. Once the visual aspect of the interface is defined, it is saved in a "figure" file. To implement the behavior of the interface, a Matlab code file is generated and linked to the figure. In this code file, we can develop the computations following a click on a button, save the text typed by the user or add some dialog boxes for instance. For the mouse-tracker, I developed two interfaces. One has a reduced size so it minimizes the influence of the interface during a usability testing (see Figure 2-left). The second one is larger and provides more information such as the real time X and Y position of the cursor (see Figure 2-right). In both cases, the 'START' and 'STOP'

buttons respectively start and stop the recording of the cursor trajectory and the ‘SAVE’ button allows to save the cursor trajectory as a matrix for further use.

Once the user starts the recording, the interface will run in the background. Indeed, for usability experiments, we do not want the interface to interfere with the environment that is being tested. At the end of the experiment, the interface comes back to the foreground to let the user stop the recording.

1.2 Technical details

To determine the time needed to complete a task and to compute the speed of the cursor, it is necessary to create a timer. Matlab offers the ‘clock’ function that gives the current date and time with a precision of 1 millisecond. When the recording of the cursor trajectory starts, a reference time T0 is stored. Then, with the function ‘etime’, we can compute the time elapsed since T0. The sampling is made at a rate of one measure every 150 milliseconds. In the literature, the sampling rate range from one measure every 100 milliseconds to one measure every 500 milliseconds. I chose the value of 150 milliseconds to be accurate enough without compromising the performance of the mouse-tracker with too much data.

However, with Matlab functions, it is only possible to get the cursor position within the created GUI. Hence, I called some JAVA functions that are compatible with Matlab and allows to get the cursor position relatively to the screen without regard to the application being used. The function used to record the cursor position imposed that the upper-left corner of the screen has the coordinates (0, 0) and the Y-axis is oriented

downward. The development was done on a 15.6” screen with a resolution of 1366 by 768 pixels. Thus, the lower-right corner has the coordinates (1366, 768).

Once the position of the cursor is known and time intervals given by the timer, it is easy to compute the velocity of the cursor along the X and Y-axes thanks to the following formulas:

$$v_x(t_i) = \frac{x(t_i) - x(t_{i-1})}{t_i - t_{i-1}} \qquad (1)$$

$$v_y(t_i) = \frac{y(t_i) - y(t_{i-1})}{t_i - t_{i-1}} \qquad (2)$$

These speeds are computed while the execution of the code is stopped to wait for the next sample so, it does not influence the performances of the mouse-tracker. Moreover, it avoids a processing phase before the data can be interpreted.

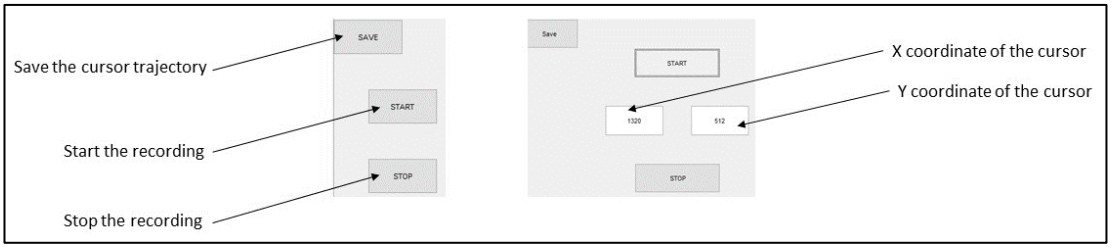


Figure 2. Interfaces for the mouse-tracker. Left: minimal interface. Right: the general interface of the mouse-tracker with textboxes to display the cursor position.

1.3 Data interpretation

The data from the recording of the cursor trajectory can be stored as a matrix in an appropriate Matlab file to be used later. In this case, the matrix contains the X and Y coordinates of the cursor, the timestamp and the X and Y-velocities. From this matrix, we can retrieve meaningful information such as the duration of the experiment, the cursor trajectory along each axis, velocities or the maximum deviation from a straight trajectory.

The field of research highly influences the interpretation techniques. For instance, psychology scientists will be interested in the deviation from a straight trajectory and the velocities to quantify hesitations (Smeding et al., 2016). On the other hand, for studies that aim at identifying pattern in cursor trajectories temporal visualizations of the positions or velocities will be privileged (Räihä et al., 2005).

The mouse-tracker's functionalities were tested in an experiment aiming at identifying cursor movements' patterns during a reading task. In Figure 1, the cursor coordinates are superimposed on the screenshot of the text read. The different colors show different patterns and the fixations on the words. Figure 3 demonstrates how these patterns can be characterized by the temporal representation of X and Y coordinates separately.

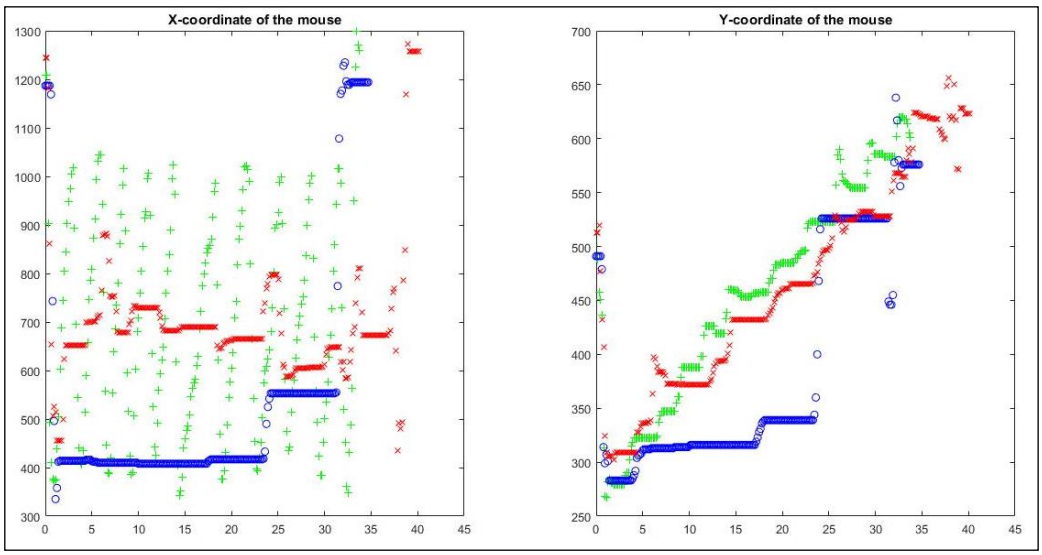


Figure 3. X or Y position of the cursor against time for three participants.

2- Define and detect patterns and cues

This second sub-section is dedicated to the different patterns and cues used to trigger the auto-completion tools and more precisely how they can be characterized by the coordinates or velocities of the cursor to be detected.

To detect certain patterns and specific cues, I gathered information from the mouse-tracking device. More specifically, the mouse-tracker computes, in real time, the position of the cursor and the velocities along each axis. If it is necessary, it is also possible to determine the angle of the movement based on two successive positions. All these information are then interpreted to identify patterns and cues that are detailed in the following sections.

2.1 Cursor position

Even if the cursor position is the most basic knowledge that can be retrieved from the mouse-tracker, it gives significant insight into the user behavior. First, it is easy to detect the presence of the cursor close to a specific point (see Figure 4). The cursor is considered close to a target point if it is included in a 15 pixels radius of this target point. To determine if the cursor at position (x, y) is around a specific point, I used the characteristic equation of a circle of radius R centered on (X, Y):

$$(x - X)^2 + (y - Y)^2 = R^2 \quad (3)$$

More generally, the position can be used to detect the presence of the cursor in specific areas of the screen rather than around a point. For example, we can be interested in the presence of the cursor at the top or the bottom of the screen. Consequently, we can define the top part of the display as the upper quarter of the screen and the bottom of the

screen as the last 20% of the screen (see Figure 5). To determine that the cursor is one of these parts of the screen it is only necessary to verify that the Y coordinate of the cursor respects certain conditions. Given a 15.6" screen with a resolution of 1366 by 768 pixels and given that the Y-axis is oriented downward, the cursor at position(x, y) being in the upper part of the screen is equivalent to the following equation:

$$y \leq 192 \qquad (4)$$

The second information provided by the mouse-tracking device is the velocities of the cursor along the X and Y-axes that derived from the position. This information is useful to detect the direction of a cursor movement. The velocity of the movement can also be linked to the user's hesitations. Indeed, a fast movement indicates that the user is confident in the target he or she is aiming at. On the contrary, slow movements can relate to either involuntary movements or a user who is not confident in the target aimed at.

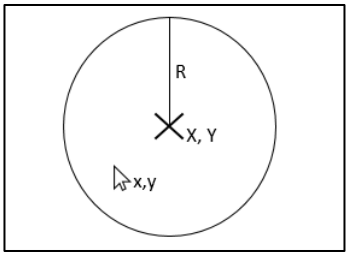


Figure 4. Detection of the presence of the cursor around a given point (X, Y).

2.2 Checkmark cue

For the purpose of the auto-scrolling tool, it was decided to use cues in the shape of checkmark pointing upward or downward (see Figure 6). These cues are easily recognizable with the help of the velocities values but have few chances to be drawn inadvertently. The cue is characterized by a positive X-velocity for both orientations and a positive then negative Y-velocity for the downward cue and vice-versa for the upward

cue. To determine this change of sign of the Y-velocity, only the last two positions of the cursor are examined, each position corresponding to one branch of the checkmark. The concept is detailed in the following algorithm:

Algorithm 1: detection of the checkmark cue

```
if no movement toward right-hand side then
    cueUp = false
    cueDown = false
else
    cueUp = boolean(negative then positive velocity along the Y-axis)
    cueDown = boolean(positive then negative velocity along the Y-axis)
end if
```

To limit the chances of unwanted detections, it is necessary to introduce some tolerance regarding the detection of the checkmark. Indeed, if a user moves the cursor along a line of text, he or she is not likely to move in an absolutely straight line and any deviation along the Y-axis could be interpreted as the cue. To tackle this issue, the velocity along the Y-axis is not compared to 0 but to the value of ± 67. This value corresponding to a movement of 10 pixels for a 150 milliseconds interval.

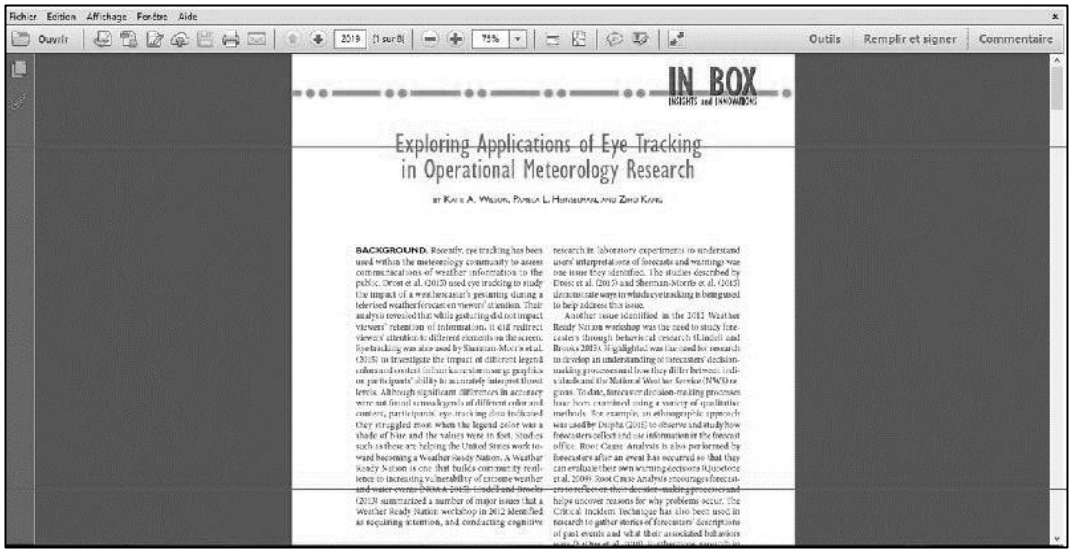


Figure 5. Top and bottom part of the screen as described earlier.

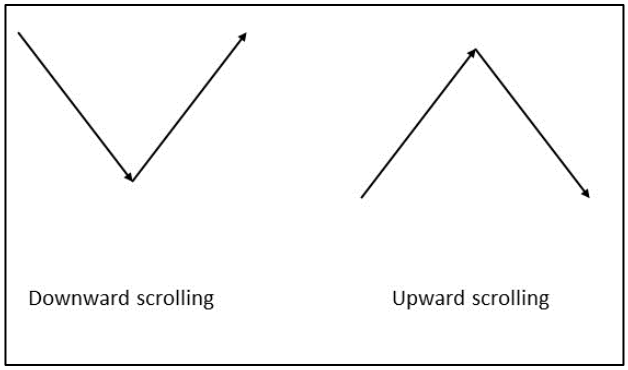


Figure 6. Checkmark cues used for the auto-scrolling tool.

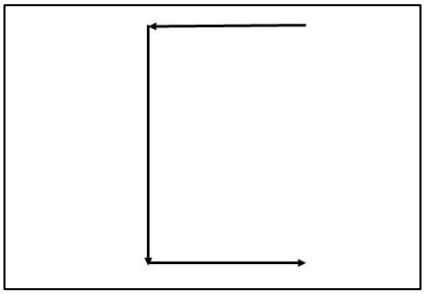


Figure 7. Bracket-shaped cue.

2.3 Bracket-shaped cue

Another cue can be detected thanks to the velocities. This cue is a bracket-like shape drawn on the touchpad (see Figure 7). It is characterized by a horizontal movement to the left followed by a downward movement followed by a horizontal movement to the right. The horizontal movement to the left (respectively to the right) can be detected with negative (respectively positive) velocity along the X-axis and the downward movement is indicated by a positive velocity along the Y-axis. We want to detect the pattern as soon as it is drawn by the user but we are unable to detect in advance when the user started the pattern. Thus, to identify the pattern, the last positions are examined in reverse order and it is necessary to reverse the order until the pattern is found or all the positions were

explored. Consequently, it is necessary to reverse the order in which each specific movement is looked for as presented in the following algorithm:

Algorithm 2: detection of the bracket-shaped cue

```
patternFound = false
startingPosition = [0; 0]
while horizontal movement to the right do
  go to the next position
end while
if more positions to be explored then
  while vertical downward movement do
    go to the next position
  end while
  if horizontal movement to the left then
    patternFound = true
    startingPosition = last point of vertical movement
  end if
end if
```

2.4 Angle of the movement

To improve the detection of movements' target and to reduce the amplitude of movement required to detect a pattern, one opportunity is to use the angle of the movement. This way, it is no longer necessary to move the cursor up to the target point but only to move in direction of this target point during several iterations. For each point of the subset of positions considered, the actual angle formed by the cursor and the point of interest will be compared to the optimal angle formed by the first point of the subset and the target (see Figure 8). The angle formed by a horizontal line and the line between two points of coordinates (x_1, y_1) and (x_2, y_2) is given by the following formula:

$$\alpha = atan\left(\frac{y_2 - y_1}{x_2 - x_1}\right) \quad (5)$$

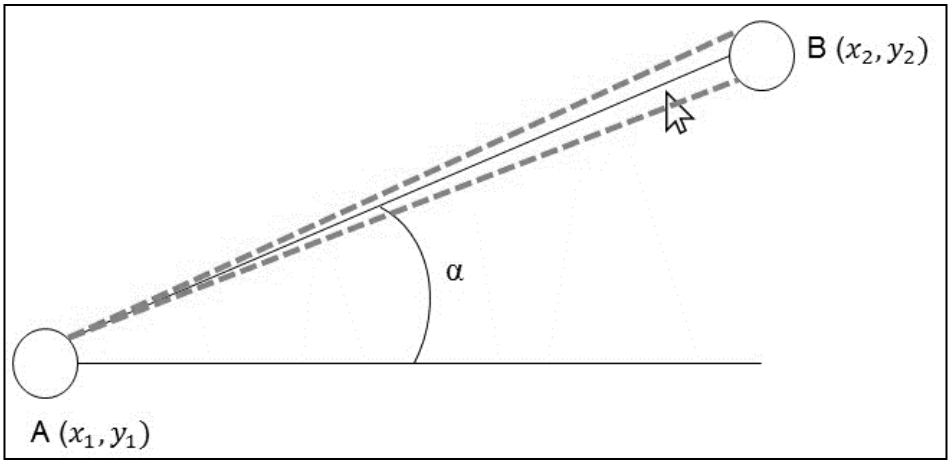


Figure 8. Illustration of the use of the movement's angle to detect the target. Point A represents the current point, point B the target point and the dashed lines represent a tolerance margin around the optimal value α.

2.5 Circular shape

To detect a circular movement, four successive angles are computed from the last five positions recorded following the formula described in the previous section (see Figure 9). If the values of the angles are increasing and the last angle is superior to 160°, then it is considered as a circle. The concept of this cue's detection is developed in the following algorithm:

Algorithm 3: detection of the circular cue

```
patternFound = false
startingPosition = [0; 0]
initialize angles values
while angle value increasing AND new value to compute do
  compute angle value
  go to the next position
end while
if last angle around 160° then
  patternFound = true
  startingPosition = approximate center of the circle
end if
```

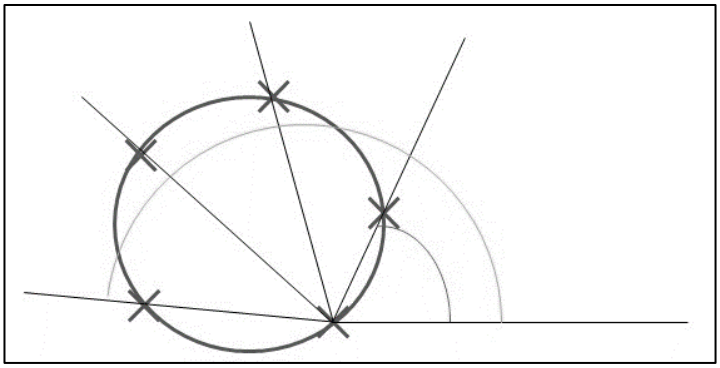


Figure 9. The different angles computed to detect a circular cue.

3- Applications

This last sub-section focuses on the tasks that are auto-completed, namely scrolling a document, selecting text and navigating on a web browser. For each task, details are given on how the auto-completion tool helps finish the task and the different patterns or cues considered to trigger the auto-completion.

To be able to create the auto-completion tools, it is necessary to simulate mouse events like clicks and scrolling but also to move the cursor over a specific point. These functionalities rely on the methods of the JAVA 'Robot' class that implements mouse events and text-typing simulation. For this research I used in particular the 'mouseWheel' function to simulate the use of the scrolling wheel, the 'mouseMove' function to shift the cursor to a given position and lastly the 'mousePress' and 'mouseRelease' functions that simulate respectively a button press or release of the given mouse button (left, right or middle button).

3.1 Scrolling a document

When we use a laptop with a touchpad, we do not have access to a scrolling wheel. Thus we need to use the scrolling bar but that requires to click on small icons that are

difficult to reach. Indeed, according to Fitts' law, the smaller the target, the longer is the time needed to reach it. This time will be even longer for older or motor-impaired people. Moreover, this task is critical as it is regularly performed when browsing the web or reading any kind of documents.

In the following sub-sections, two different approaches are discussed but they share some common ground. In both cases, the interface remains the same than for the mouse-tracking tool (see Figure 2-right); only the code related to the interface was modified to include the pattern detection and automatic scrolling. This tool can be used on any application ran on the computer (e.g. reading a PDF file, editing text, browsing a web page) once the interface is started and, when the cue is detected the display is scrolled for a value of 5 notches (the unit that describes the scrolling wheel movement) that, in most cases, transfers the text that was at the bottom of the page, up to the top of the screen (see Figure 10).

3.1.1- Position-based solution

As discussed in the literature review, it appears that in the case of search engine results page, cursor and gaze are aligned and thus, the position of the cursor is a good indicator of where the attention of the user is focused. Then we can extrapolate that when one is reading a document, the position of the cursor can be used to determine when the reader reaches the end of the document on the screen and then trigger the automatic scrolling. Thus, this first alternative is entirely based on the position of the cursor on the screen. More precisely, I considered that, if the user moves the cursor downward from the top to the bottom of the screen, it means that he or she is browsing the content of the

page and is most likely to want to continue downward. The operating principle for this pattern detection is detailed in the following algorithm:

Algorithm 4: automatic scrolling using the cursor position

```
if cursor in the upper zone then
  isInTop = true
end if
if isInTop then
  compute velocity along the Y-axis
  if positive velocity along the Y-axis and cursor at the bottom of the
  page then
    isInTop = false
    pause for 2 seconds
    scroll down 5 notches
  end if
end if
```

Whenever the user starts using this tool, the code enters a loop to look for the pattern and automatically scroll down the page when the pattern is detected. The loop is exited when the user presses the 'STOP' button. In more details, at each iteration, the position of the cursor is examined. If it's in the upper zone (as described in the 'Cursor position' sub-section), the velocity along the Y-axis is computed. If this velocity is positive, it indicates a downward movement. Then, after one or more iterations, if the cursor reaches the bottom part of the screen, the pattern is considered complete. Then, execution is paused for 2 seconds to let the user finish reading after the cursor moved to the bottom of the screen. Finally, a scrolling equivalent to 5 notches is simulated. It is important to note that for this tool, velocities are not computed automatically as the detection is very simple and it would require unnecessary time and space.

3.1.2- Cue-based solution

The previous solution is easy to understand and implement. However, one drawback emerges as not every reader uses the cursor as a reading aid. Thus, if the cursor

stays still, it is impossible to detect that the reader reached the bottom of the page and wanted to scroll the page to continue reading.

This second alternative does not rely on the cursor position but on specific cues that trigger scrolling (see Figure 6). Indeed, I wanted to include more flexibility than in the previous solution in particular, by adding an upward scrolling. In this case, the user can draw a checkmark pointing upward or downward on the touchpad and the corresponding scrolling movement is automatically performed. The mechanics of this tool is detailed in the following algorithm:

Algorithm 5: automatic scrolling using the checkmark cue

```
[cueUp, cueDown] = findCue(positions)
if cueUp = true then
  cueUp = false
  scroll up 5 notches
else if cueDown = true then
  cueDown = false
  scroll down 5 notches
end if
```

3.2 Selecting text

This second tool aims at simplifying text selection to copy, delete or modify some text. Indeed, I considered the case where the user only has access to a touchpad. Thus, it can be hard for some people to maintain the left button of the mouse pressed while moving the cursor to select text. In this solution, a cue drawn with the touchpad will automatically simulate the left button press. Then the user has his or her hands free to select the text he or she is interested in.

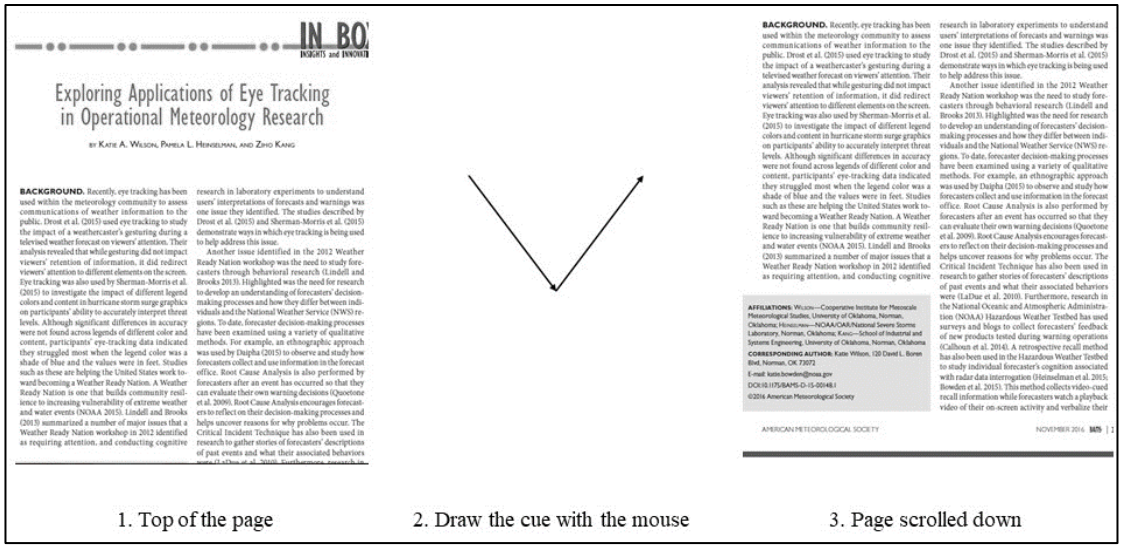


Figure 10. Successive steps to automatically scroll the page with the cue detection.

In the following sub-sections, three different cues are discussed but in all solutions, the interface of the mouse-tracker remained the same and only the code was modified to implement the appropriate pattern detections. Similarly to the previous tool, when the interface is started, at each iteration, the pattern is looked for among the last positions recorded. Once the pattern is detected, it is necessary to determine where the text selection begins as drawing the cue to trigger the assistance may entail moving the cursor around the first word of the selection. Once the position of the first word is computed, the cursor is automatically moved to that position and a left button press is simulated. The left button is released as soon as the velocities along the X and Y-axes are lower than 0.75 pixel per second. This value allows the user to precisely select some text with slower movements while accounting for involuntary movements when the user leaves the cursor still at the end of the selection.

3.2.1- Bracket shaped cue

The first possible cue that was considered to trigger these actions is a bracket-like shape drawn on the touchpad (see Figure 11). When the pattern is detected, the cursor is moved to the position of the vertical bar of the bracket with an offset of 5 pixels to the left. This offset is introduced to take into account a lack of precision from the user as it can be difficult to draw the bracket exactly at the beginning of the first word. This way, the offset can add a couple characters to the selection if the bracket was not placed at the exact position. At each iteration, the cue is looked for among the last 10 positions recorded (or all the positions if less than 10 were measured).

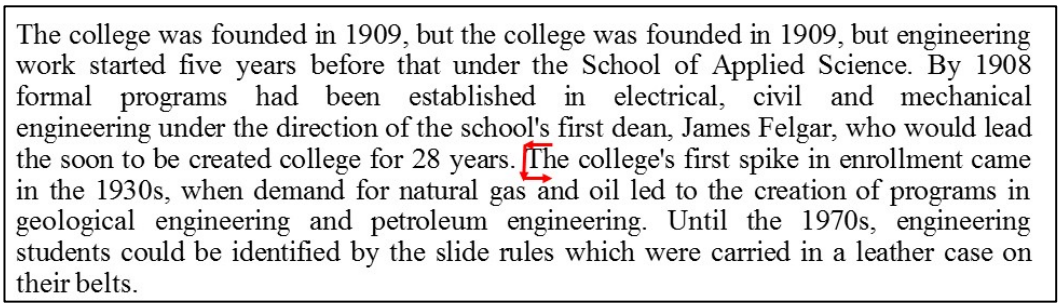


Figure 11. An example of how the bracket-shaped cue needs to be drawn around text.

3.2.2- Circular cue

To be used efficiently in the text selection assistance tool, the bracket-shaped cue needs to be drawn with precision and with a limited size which may entail difficulties for people lacking motor control. To cope with the precision demanded by the previous cue, another idea come up and consists in using a circular shape to circle the beginning of the text to be selected.

In this case, the beginning of the selection is set to the approximate center of the circle drawn by the user. To determine the center, the position related to the angle that is

the closest to 160° is retrieved. This point can be considered diametrically opposite to the first point. Then the position of the center is computed as the middle of the segment between these two points. The selection stops with the same conditions than the bracket-shaped cue solution.

Whereas this cue should be easier to draw and be less precision-demanding with regard to the beginning of the text selection, this solution suffers from a major drawback as the method employed to detect the circular cue is not restrictive enough and entails many unintended detections that hinder the navigation on any type of document.

3.2.3- Checkmark cue

The last cue that was considered is the checkmark shape, similar to the scrolling tool, but this time, pointing toward the left. As described in section 2.2, this pattern is detected by examining the last two positions in the cursor trajectory thus, the starting point for the selection is inferred from these two positions. To start the selection, the cursor is moved to the X coordinate of the second position (corresponding to the right extremity of the checkmark) and the Y coordinate of the first position which approximates the middle of the cue (see Figure 12). The selection stops with the same conditions than the previous solutions presented.

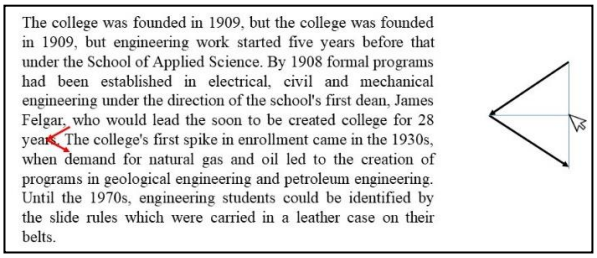


Figure 12. Left: an example of how the checkmark cue needs to be drawn around text. Right: explanation on where the cursor is moved for the beginning of the text selection.

3.3 Navigate on a web browser

Motor-impaired people can encounter difficulties browsing the web on a search engine. Indeed, using the different features of a search engine often entails moving the cursor across the screen. To tackle this issue, two solutions with different features were explored to offer an automatized movement of the cursor to specific points of interest. For both solutions discussed in the following sub-sections, the interface of the mouse-tracker has been modified. Indeed, to make these tools operable on any computer, it is necessary to take into account different screen resolutions that lead to different coordinates for the points of interest.

The original mouse-tracker interface has been modified (see Figure 13). Several editable text boxes were added to enable the participant to enter the X and Y coordinates of the different points of interest on the Google homepage. This way, the code can be executed on any computer without regards to the screen resolution.

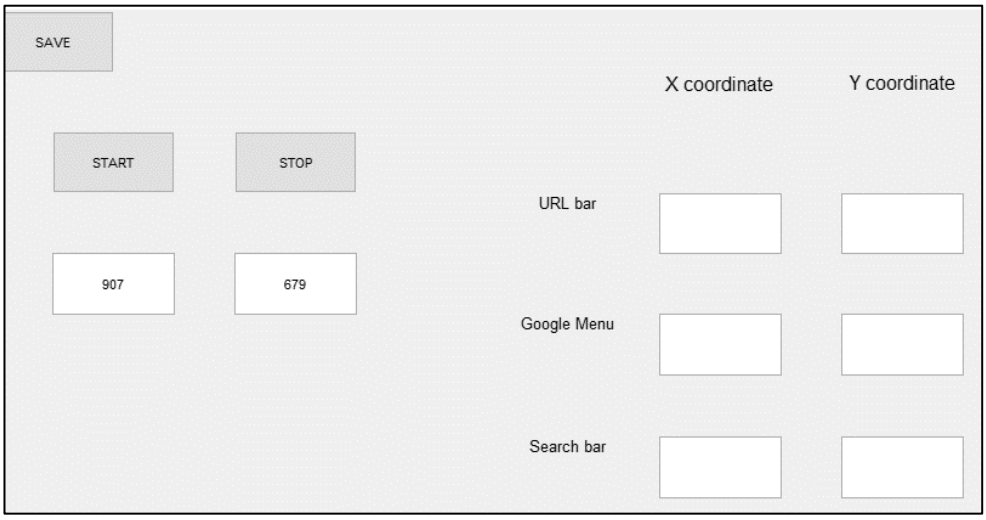


Figure 13. The interface created for the navigation improvement tools.

3.3.1- Opening a Google menu tab

This first example focuses on opening a tab of the Google menu which requires often time a wide movement to reach the menu icon and to click precisely on a small icon surrounded by several other icons. For the demonstration, the tab opened by this tool is Google News but any tab can be opened provided that we know its coordinates. In this example, the user starts from the Google search bar and moves toward the Google menu to open Google News. The pattern recognition refers to the movement from the search bar to the menu. Then the auto-completion includes moving the cursor to the center of the Google menu, simulate a click, move the cursor to the Google News icon and simulate another click (see Figure 14).

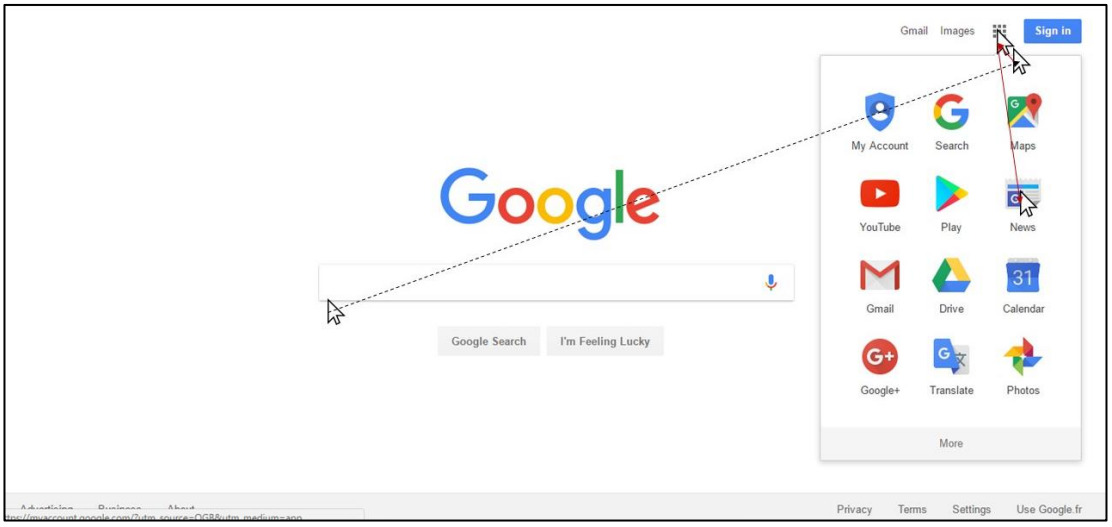


Figure 14. Steps for the automatic opening of Google News. The user moves from the search bar to the menu (dashed line) and the tool completes the action (red line).

I developed two methods to detect this pattern. The first method only uses the position of the cursor and detect the presence of the cursor around the two points of interest that are the search bar and the Google menu. Each time the mouse-tracker gets a

measure of the cursor position, the predefined pattern is looked for among the last 20 positions recorded (or all the positions if less than 20 were measured). The concept for the pattern's identification is as follows:

Algorithm 6: pattern detection for Google News using the cursor position

```
patternFound = false
if last position around Google menu then
  while more positions to be explored do
    if position close to the search bar then
      patternFound = true
    else
      go to the next position
    end if
  end while
end if
```

The second method implements the angle of the movement. Indeed, once the cursor is detected around the search bar, the angle of movement is analyzed and compared to the optimal angle between the search bar and the Google menu. Then, if three measures show that the angle of movement is within 10° of the optimal angle, the pattern is considered complete.

Algorithm 7: pattern detection for Google News using angles

```
patternFound = false
if first position around the search bar then
  compute optimal movement's angle
  for all remaining positions do
    compute movement's angle
  end for
  if all angles within 10° of the optimal angle then
    patternFound = true
  end if
end if
```

Once one of these patterns is detected, the auto-completion starts. The cursor is moved at the center of the Google menu button, a click is performed which opens the

menu then, the cursor is moved toward the Google News icon and another click is performed to open it (see Figure 14).

3.3.2- Complete recurring movements

The previous tool proved that identifying movement toward a specific point is possible. However, its scope is relatively narrow as the detection is only possible if the user moves from a given point to the target and the auto-completion only include one feature when we can imagine many more.

To improve the opportunities offered by the previous tool, a second tool has been developed by dividing the Google homepage into four quadrants that contain different points of interest. The upper-left quadrant contains the URL address bar, the upper-right quadrant includes the Google menu and the lower-left quadrant gives access to the search bar. Then, I identified what are the movements that the user is the most likely to perform and which require a large amplitude of movement to complete them automatically (see Figure 15). This way, the movement is identified from a quadrant to a target point which is less restrictive and three different patterns are considered in the same tool.

Once this tool is started, the position of the cursor is recorded and at each iteration, depending on the quadrant in which the cursor is currently located, specific patterns are looked for and if one is detected the movement is automatically operated by moving the cursor to the corresponding point of interest as described in the following algorithm:

Algorithm 8: pattern detection for navigation improvement

```
toURLBar = false
toSearchBar = false
toMenu = false
if four new positions to analyze then
  if cursor in the upper left quadrant then
    toSearchBar = findSearchBarPattern(positions)
    if toSearchBar = false then
      toMenu = findMenuPattern(positions)
    end if
  else if cursor in the lower left quadrant then
    toMenu = findMenuPattern(positions)
    if toMenu = false then
      toURLBar = findURLPattern(positions)
    end if
  else if cursor in the upper right quadrant then
    toSearchBar = findSearchBarPattern(positions)
    if toSearchBar = false then
      toURLBar = findURLPattern(positions)
    end if
  else
    toMenu – findMenuPattern(positions)
    if toMenu = false then
      toURLBar = findURLPattern(positions)
    end if
  end if
end if
completeToURLBar(toURLBar)
completeToSearchBar(toSearchBar)
completeToMenu(toMenu)
```

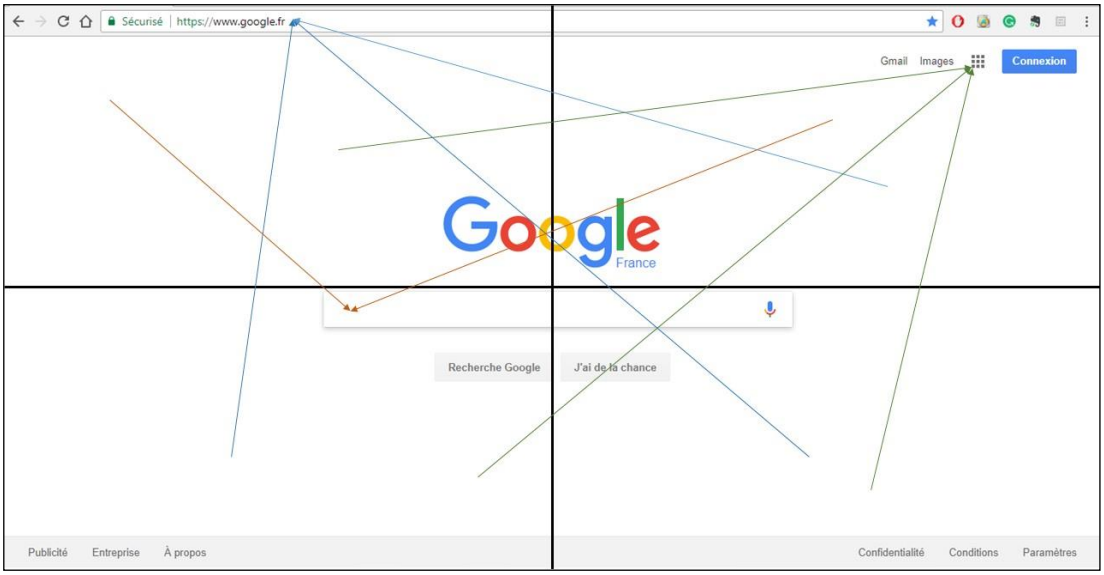


Figure 15. The homepage divided into four quadrants and the movements that can be identified and autocompleted (colored by target point).

To detect the said patterns, I used both the angle of the movement and the distance to the target. Indeed, the angle of the movement does not indicate if the cursor is moving toward or away from the point of interest. Thus, introducing the notion of distance from the target helps reduce unwanted detections by simply verifying that the distance between the cursor and the target is decreasing. At each iteration, the last four points of the trajectory are considered; the optimal angle of movement is computed as the angle between the first of the four points and the potential target and angles and distances are calculated for the next three points. Then, the pattern is considered complete if the three measures show that the angle of movement is within 15° of the optimal angle and if the distance to the target decreased. The concept for this pattern detection is the following:

Algorithm 9: detection of movement toward target point

```
patternFound = false
compute optimal angle
compute maximum distance to the target point
while new position to analyze do
  compute movement's angle
  if angle NOT within 15° of the optimal angle then
    stop computations
  end if
  compute distance to the target point
  if cursor NOT closer of at least 5 pixels then
    stop computations
    go to next position
  end if
end while
if movement toward target point for all positions then
  patternFound = true
end if
```

3.4 Final deliverable

The final deliverable takes the form of an application that combines all the tools described here in one, more complete, tool to provide an optimal experience to the user. This way, while we are browsing the web we would have assistance to move to specific

points of interest, scroll the web pages and select text. Moreover, we can imagine that the interface lets the user choose which tools he or she wants to use (see Figure 16). Thus, if the user wants assistance for scrolling and text selection for other types of documents, he or she would not be disturbed by the target identification tool.

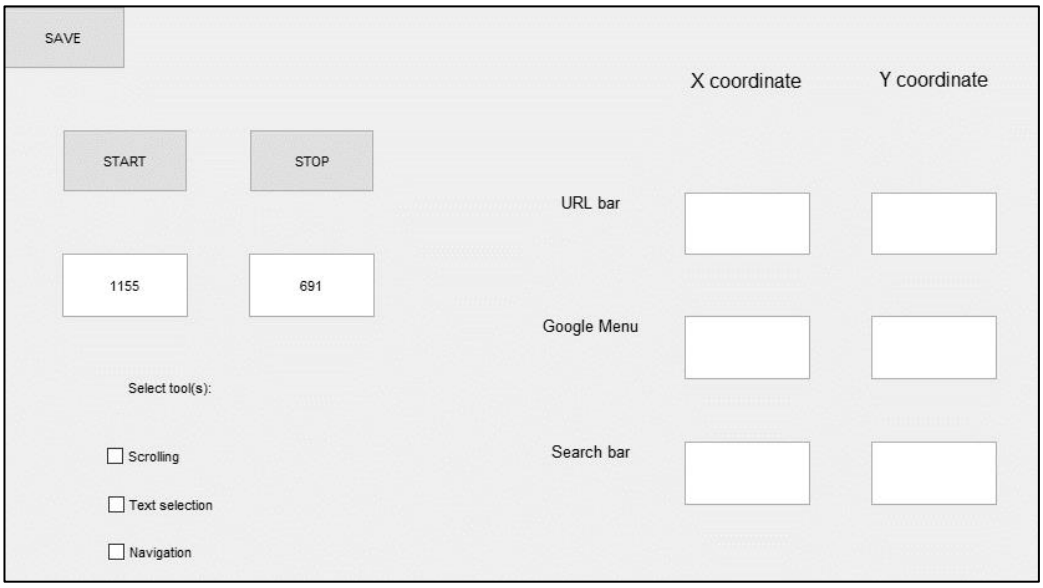


Figure 16. Interface of the final auto-completion tool.

Chapter 4: Experimentation

1- Recruitment

All participants were recruited through emails at the University of Oklahoma. They were aged 22 to 72 years old. All of them were fluent in English and able to perform simple computer tasks involving the touchpad. Participants were split into two groups: a control group of participants (aged 23 to 72 years old, mean = 34.43, sd = 13.01) who had no access to the auto-completion tools and whose tasks completion time served as a reference and an experimental group (aged 22 to 57 years old, mean = 28.46, sd = 9.48) who had access to the auto-completion tools. They were randomly assigned to either group at the condition that there was an equal number of participants in each group. Before starting the experiment, they were asked to sign a consent form authorizing the recording and analysis of the cursor movements during the experiment and the responses to a usability questionnaire.

2- Technical details

To compare the completion time of each task, it is necessary to be able to detect when a given task is completed by the user. For participants that were part of the experimental group, it was easy as the completion of the task corresponds to the end of the auto-completion code execution. But for the control group, it required additional computations. In the case of the navigation on Google, it is possible to detect the presence of the cursor around one of the point of interest which means that the task involving to move the cursor to this point is complete. However, there is no particular event related to

the completion of the scrolling or text selection tasks. Thus, it was impossible to test all these tools in one scenario and still get a precise completion time.

3- Navigation improvement tool

The interface used for this part of the experiment is identical to the interface described in Section 3.3 (see Figure 13). Once all the required positions were entered, participants clicked the ‘START’ button. After that, the recording of the cursor position started along with pattern detections. To have all participants start in the same conditions, a new web browser page was opened on the Google homepage and the cursor was automatically moved to the position (1300, 700). This position corresponds to the lower right quadrant and was chosen to maximize the distance to cross to complete the first task of the scenario. The scenario included the following steps:

1. Move the cursor to the URL bar and enter the URL: https://en.wikipedia.org/wiki/University_of_Oklahoma.
2. Go back to the homepage and move the cursor to the search bar to enter the request ‘Wikipedia OU’.
3. Open the Google News tab from the Google menu.

Once the participants completed all the steps, the recording was stopped by clicking the ‘STOP’ button and the participants were asked to save the data in a file for further analysis.

4- Text selection tool

This tool displayed a smaller interface as the screen resolution has no influence on the code (see Figure 2-Right). Still, the original code was modified so that when the user launched the application, the system web browser opened at the Wikipedia web page for the University of Oklahoma. Once the recording was started, the only task to be performed was to select the first paragraph of the article. Then, participants stopped the recording and saved it for further analysis.

5- Scrolling tool

The interface used here was the same than for the text selection tool. Once again the University of Oklahoma Wikipedia web page opened automatically. When participants started the recording, the task to be performed consisted in scrolling down to the first section and back up to the top of the page. Then, the recording was stopped and the data were stored in a file for further analysis.

6- Usability questionnaire

When all these tasks were performed, participants were asked to answer several questions (see Appendix A) aiming at quantifying how difficult these tasks were perceived on a five-point scale and the user's familiarity with the touchpad (all participants) and how easy it was to use the auto-completion tool on a five-point scale (only participants from the experimental group).

These questions were established to evaluate several usability measures as described in the following table:

Usability measure	Group	Questions
Efficiency	Both	• In general, I find it hard to use the touchpad • I found the tasks hard to perform
Learnability	Experimental	• It is easy to understand how auto-completion tools work
Satisfaction	Experimental	• Evaluate how useful was the auto-completion tool • The navigation was harder due to the tools

Table 1. The different questions of the usability questionnaire classified by the usability measure they evaluate.

Chapter 5: Results

All mouse-tracking data were processed using Matlab to retrieve each task's completion time. When a participant was not able to trigger auto-completion tools or did not leave the cursor still around target points, it was impossible to generate automatically the completion time. Thus, some data were analyzed manually to determine as precisely as possible the completion time. Across all participants and tasks, three values were missing, one due to data corruption and two other because it was impossible to determine an exact completion time. Once all completion times were determined, the data were analyzed using the language R to perform data visualizations and statistical analysis summarized in the following sections. ANOVA analysis of the participants' age of both groups showed no significant difference between control and experimental group $(p - value = 0.74)$ so age has not influenced the results between the two groups.

1- Familiarity with the touchpad

This factor was evaluated using a yes or no question on whether participants found it difficult to use the touchpad in general. 26% of the participants replied yes indicating that using the touchpad is not perceived as difficult for most people.

2- Perceived difficulty of the tasks

This measure is an aggregate value on how participants perceived the difficulty of the tasks. A lower score indicates that the tasks were easy. The score attributed by participants in question 1 are presented in the following histogram. We can see that scores range from 1 to 5, indicating that all participants had not the same perception of the

difficulty of the tasks. The mean score is 1.96 which tends to show that overall, the tasks were not perceived as difficult.

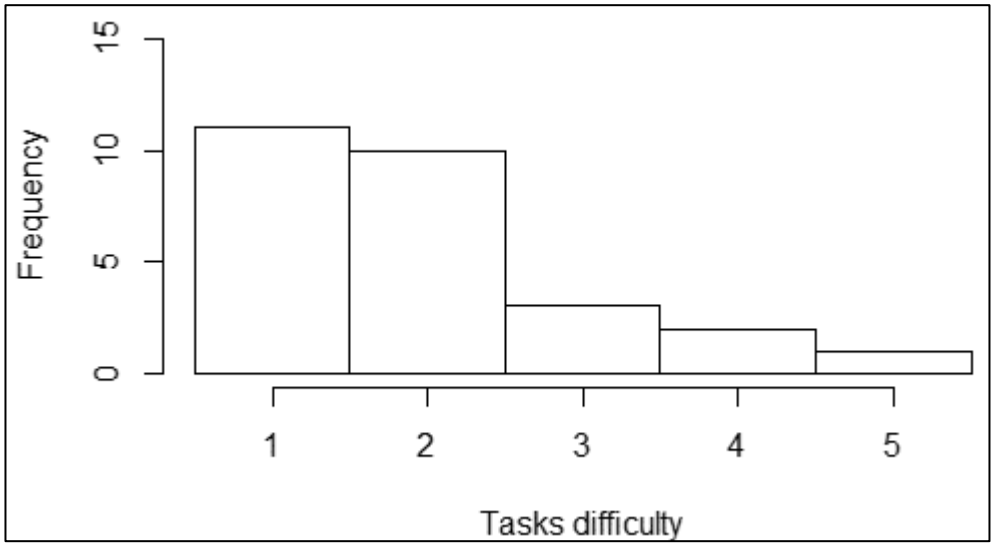


Figure 17. Histogram presenting the score given for tasks difficulty.

If we take a closer look at these scores by participants' group, it appears that 71% of the control group gave a score of 1 meaning that the tasks were easy but experimental group participants gave in average a score of 2.69.

3- Intuitiveness of the patterns

The score attributed by participants in question 2 are presented in the following histogram. A higher score indicates that it was easy to understand the cues and patterns detected for the auto-completion tools. We can see that scores range from 2 to 5, indicating that the cues decided to trigger auto-completion tools were easy to understand. This factor is all the more important as the intended target for these tools are elder people who can suffer from memory impairment. It is highly important that the patterns used to

trigger the auto-completions are easy to memorize and the mean score of 4.46 shows that there is no need to improve the intuitiveness of these patterns.

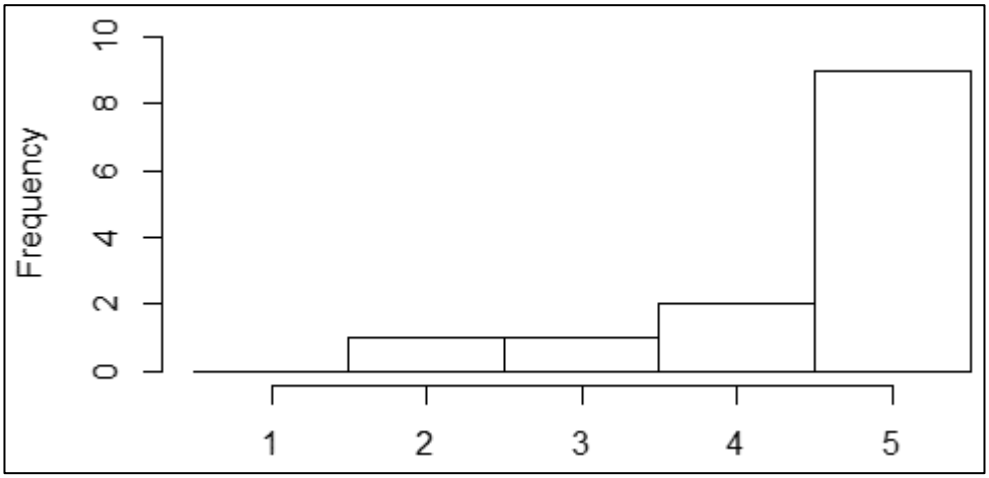


Figure 18. Histogram presenting the score given for patterns intuitiveness.

4- Auto-completion tools usability scores

The usability of each auto-completion tools along with an overall appreciation of all the tools was assessed by the participants on a 5-point scale with a score of 1 indicating dissatisfaction and a score of 5 meaning that the participant was satisfied. The following figure displays the median and first and last quartiles of the scores in boxplots.

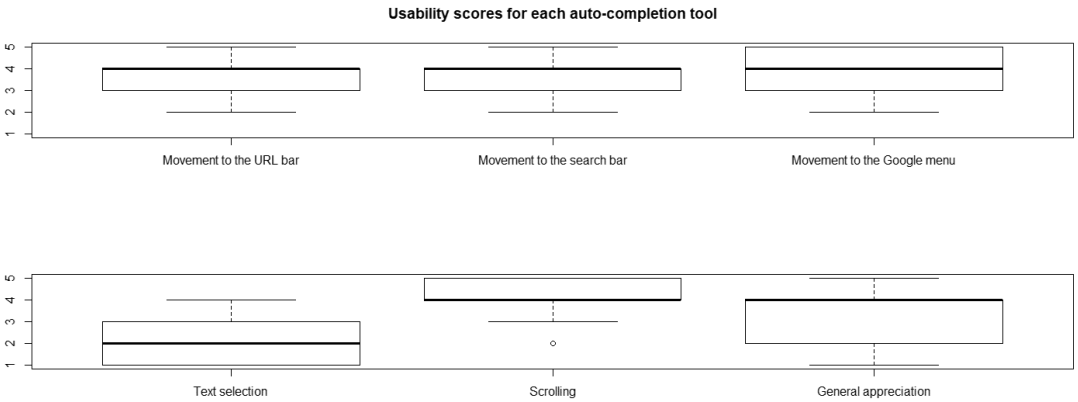


Figure 19. Boxplots of the usability scores given for each auto-completion tool.

The general appreciation score is 3.31 on average indicating that the tools are considered helpful. It is particularly true for the scrolling assistance tool that reached a 4.08 average score with 46% of the participants giving a score of 5. On the other hand, the text selection assistance tool scored poorly with an average score of 2.31 and 54% of the participants giving a score of 1 or 2. Regarding the navigation improvement tool, the mean scores are all over 3.6 which is satisfying but not as high as the scrolling assistance tool.

5- Influence of the tools on the computer use

The score attributed by participants in question 9 is presented in the following histogram. A lower score indicates that the tools improved the use of the computer. We can see that the scores range from 1 to 5 with a mean of 2.92, indicating that the navigation is slightly improved thanks to the tools. However, this value is close to 3 which is the intermediate value in the 5-point scale used in the questionnaire. Thus, it is hard to conclude definitively that the tools have an influence on the general use of the computer.

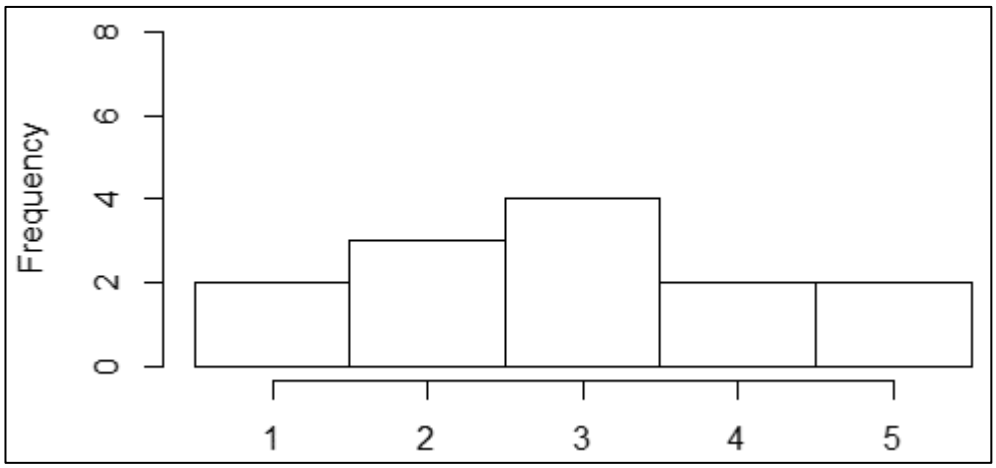


Figure 20. Histogram presenting the score given for the influence of the tools on the navigation.

6- Comparison of the completion times

All tasks of the experiment are discussed separately in the following sections. It is important to note that outlying values corresponding to unexpectedly high completion times were removed. Indeed, 26% of the participants declared that they find it difficult to use the touchpad in general so the difficulty they encounter with the touchpad may have slowed them in the completion of the tasks involved in the experiment. To compare both groups, I performed an ANOVA to determine if the time required to complete a task is significantly different from one group to the other.

6.1 Movement times

To complete the tasks of navigation on the Google homepage, experimental group participants were not required to learn how to draw a specific cue. They only had to try to complete the cursor movements following the straightest possible trajectory to trigger the auto-completion tools.

The comparison of the completion times showed that participants from the experimental group were significantly faster to move the cursor to the URL address bar ($p - value = 7.37 \cdot 10^{-3}$) and to the search bar ($p - value = 5.26 \cdot 10^{-3}$). These results appear clearly on the mean and standard error plots in Figure 21.

To complete these results, I used a Fitts' law based on the results presented in MacKenzie et al. (1992). For pointing and selecting, the equation giving the completion time is:

$$MT = 53 + 148 \cdot ID \qquad (6)$$

where ID represents the index of difficulty of the task, computed as:

$$ID = \log_2\left(2 \cdot {}^{A}/_{W}\right) \qquad (7)$$

with A representing the distance to the target and W its width.

We can notice that all minimum movement times are lower than predicted except for the movement to the Google menu for the control group. Moreover, the minimum movement times for the experimental group are almost twice lower than the theoretical movement time. Finally, if we consider only the participants who were able to trigger auto-completion tools, their mean movement time are lower (0.536s, 0.575s and 0.616s respectively).

	A (px)	**W (px)**	**ID**	**MT (s)**
Movement to the URL bar	354	15	5.56	0.876
Movement to the search bar	429	15	5.84	0.917
Movement to the Google menu	870	15	6.86	1.068

Table 2. Theoretical movement time resulting from Fitts' law.

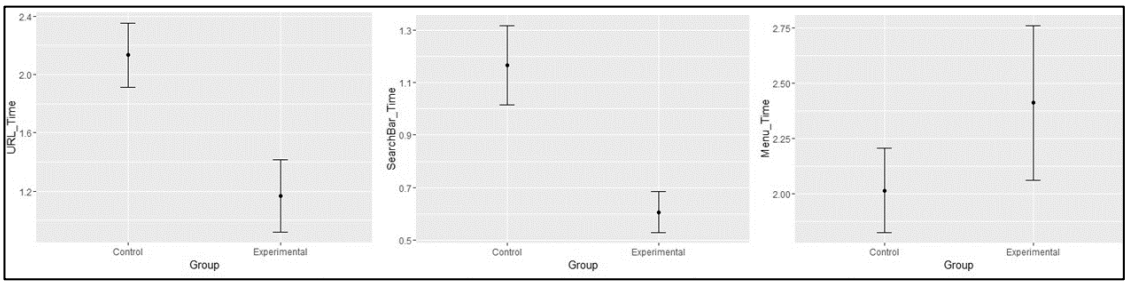


Figure 21. Mean and standard error plots for each movement completion time, by group, in seconds. From left to right: movement to the URL address bar, movement to the search bar and movement to the Google menu.

6.2 Text selection task

For this task, participants from the experimental group had to learn how to draw the cue that triggers the assistance tool. More precisely, they had to go through a trials and errors phase to draw the cue accurately at the beginning of the text they had to select, and they also had to move the cursor fast enough to prevent the tool from releasing the left button. This training phase was included in the completion time which explains why the experimental group needed significantly more time to complete the text selection task ($p - value = 1.27 \cdot 10^{-11}$).

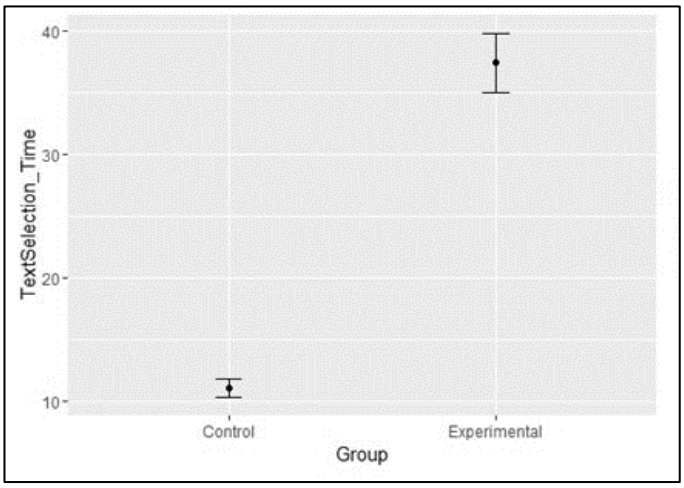


Figure 22. Mean and standard error plots for text selection task completion time, by group, in seconds.

6.3 Scrolling task

Again, for this task, participants from the experimental group had to learn how to draw the cue that triggers the assistance tool. More precisely, they had to go through a trials and errors phase to draw the cue accurately. This training phase was included in the completion time which explains why the experimental group needed significantly more time to complete the scrolling task ($p - value = 3.46 \cdot 10^{-2}$).

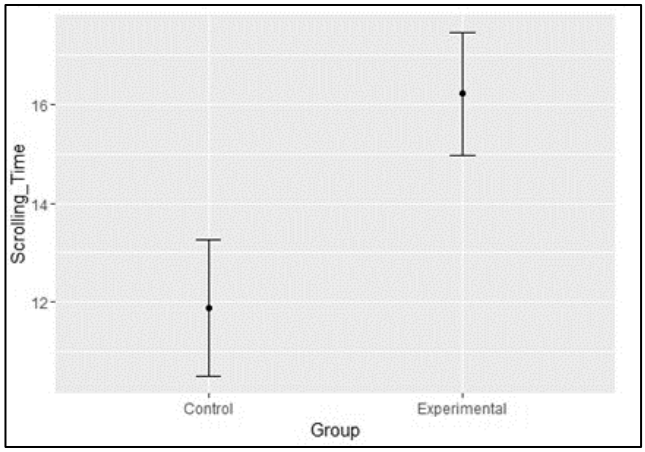


Figure 23. Mean and standard error plots for scrolling task completion time, by group, in seconds.

7- Comparison of the completion rates

The completion rate is defined as the ratio of participants that successfully completed the required tasks. For the experimental group, a task is considered completed if the participant was able to trigger the auto-completion tool with the appropriate pattern (see Table 3).

	Completion rate for the experimental group
Movement to the URL bar	54%
Movement to the search bar	85%
Movement to the Google menu	8%
Text selection	69%
Scrolling	100%

Table 3. Summary of the completion rates for each task for the experimental group.

We can notice that for the scrolling task, the completion rate was 100% meaning that all participants from the experimental group were able to draw the cue accurately. The movement to the search was automatically completed for 85% of the participants which is satisfying. However, for the movement toward the Google menu, it dropped to 8%.

Chapter 6: Discussions

In this chapter, the results of the experiment are discussed to explain the probable reasons behind outlying results and the main limitations of this study. The results of the experiment showed that all participants are familiar with the touchpad and most of them do not encounter difficulties using it. Moreover, the perceived difficulty of tasks involved in the testing is low. Participants from the experimental group also described the pattern and cues as easy to understand. All these indicators explain the high usability scores for the navigation and scrolling tasks as well as minimum movements times that are almost half the predictions from Fitts' law for the movement to the search bar and the Google menu. Nevertheless, some results are less satisfying than expected or need some qualifications.

First of all, regarding the perceived difficulty of the tasks, we can notice that the experimental group gave higher scores with an average score more than twice as high as the control group. This is all the more intriguing as the tasks involved in the experiment are common tasks that are performed on a daily basis by any computer user. However, we must bear in mind that participants from the experimental group needed to understand and get used to the auto-completion tools that they had to use. In particular, for the text selection and scrolling task, they had to manage to draw the cues correctly, with enough precision and speed to be detected correctly. Thus, these participants might have encountered more difficulties to perform these tasks due to the novelty of the tools. This argument is supported by several participants that commented that after some training they would be likely to be able to use the auto-completion tools more efficiently.

Secondly, some qualifications need to be drawn about the usability scores attributed to the auto-completion tools. Indeed, in the case of the navigation on Google, several participants commented that they were not able to perceive the auto-completion part, most likely because they were not able to trigger the auto-completion. Moreover, some participants did not move the cursor following a straight trajectory but made a curvier movement which prevented the detection of the movement patterns and the triggering of the related auto-completion. This is particularly true for the movement to the Google menu where only 8% of the participants triggered the auto-completion tools. Both of these issues can explain why the usability scores are not higher. Additionally, even though the completion rate for the text selection task is satisfying with 69% of the participants triggering the text selection assistance tool, the average usability score of 2.31 remains low. This is likely explained by the precision demanded by this tool to draw the cue accurately at the beginning of the text to be selected and to move the cursor fast enough to reach the end of the text before the assistance is stopped. To be able to complete this task, successful participants had to make several attempts before controlling all the parameters involved in the cue detection. Thus, without training, this solution seems to make it harder to perform a basic task which entails the lower usability score. However, it is important to note that several participants expressed a strong interest in the idea of not having to hold the left button of the mouse pressed while moving the cursor to select text. These comments confirm that the idea of simulating the left button press can be highly beneficial to improve the touchpad's usability, but some improvements are required on the cue that triggers this assistance.

The fact that participants needed to get used to the cues for the text selection and scrolling tools has also an influence on the completion times. The results showed that for both text selection and scrolling, the experimental group required significantly more time to complete the tasks than the control group. Indeed, few participants were able to trigger these auto-completion tools on their first attempt thus longer completion times. But these results must be considered carefully as the completion times take into account all the attempts made as a training. Once the potential users get used to drawing these cues, we can expect that they will be able to draw the cues precisely at the first attempt thus reducing the completion time. Moreover, if we focus on the automatic scrolling tool, it is important to note the completion rate of 100% showing that all participants were able to control this tool within seconds and more importantly, the usability score of 4.08 confirms that this tool is easy to use.

Additionally, one may argue that the completion times for the text selection and scrolling tasks are quite high even for the control group with no participant under 10 seconds. But, for all tasks, participants had to press the 'START' button to record the cursor trajectory and start the tools, reduce the interface to work on the web browser and at the end reopen the interface and press the 'STOP' button to end the recording of the cursor trajectory. But contrary to the navigation tasks where it possible to isolate the movements time from the manipulation of the interface, there is no event that can help distinguish the selection or scrolling phase. This likely explains why the completion times for these two tasks appear to be abnormally high.

On the other hand, it is necessary to consider the limitations of this study and how the results could be improved by overcoming these constraints. The main limitations in

this work come from how the GUI tool works in Matlab. Even if it simplifies the process of creating a GUI, it lacks more generality in the event handling. Indeed, Matlab provides numerous useful event handlers but they are only working when the cursor moves within the GUI. Thus to provide auto-completion tools that can work on any application running on a computer, it is impossible to use Matlab's event handlers, and it is necessary to fall back on external functions, namely using Java. But this trick is limited as it is not possible to use all Java functions in Matlab and in particular, I was not able to use the 'MouseListener' class and its functionalities. This issue mostly reduces the quantity of information we can collect to define and detect a pattern in the user's cursor movements.

Additionally, regarding the navigational tasks being tested, the tools were implemented in such a way that each subtask is flagged, which makes it easier to determine the completion time during post hoc analysis. More precisely, for the experimental group, when a pattern is auto-completed, a specific number is entered in a dedicated column of the record and for the control group, the flag is set when the cursor is around any of the points of interest. However, some participants failed to trigger the auto-completion tools or, in the case of the control group, clicked in the URL bar or search bar outside the area around the points of interests. They were still able to complete the tasks but they were not flagged during the execution of the code. Then, it was necessary to manually compute the completion times by browsing the record to spot a position that was consistent with the completion of a given task. This process lacks precision as it may introduce human error but also because participants sometimes moved the cursor to different (close) locations that are all potentially indicative of the task completion.

Chapter 7: Future work

The first area of improvement concerns the code that has already been written. More specifically, the detection of a circle that has been considered for the text selection assistance tool. The current solution as it is leads to many involuntary detections that hinder the navigation on any kind of application. Moreover, the computations to determine the beginning of the selection only gives an approximate position. Thus, it would be interesting to investigate further how we can detect a circular movement of the cursor with sufficient restrictions to avoid multiple unwanted detections. Another beneficial improvement can originate from the opportunity to use clicks information. It would enable more precise detections for existing tools but also offer new detection possibilities that will convert into new tools. The tools emanating from these new opportunities will either include longer sequence of events to provide auto-completion for more complex tasks or offer a way to differentiate patterns that may have the same movement characteristics but different clicking patterns. Eventually, we can imagine to introduce machine learning in the existing code to detect more general patterns like how the user repeats some sequence of task regularly. Another idea within the scope of the existing tools is to record which tab of the Google menu is clicked the most and once the movement toward the menu is detected, not only move the cursor to the menu but also click it and open the tab that is the most likely to be opened.

Another area of improvement involves the experiment protocol. Indeed, the scenarii used to test all the auto-completion tools only include one straightforward execution of each task. However, it would be interesting to test these tools for a longer period to let the participants use the tools in conditions that are the closest to the everyday

use. In particular, it would be highly beneficial to let participants navigate on their web browser with the auto-completion tools activated and evaluate how often involuntary pattern detections happened and the disruptions that it entailed. Moreover, as some participants declared, using the tools for a longer time may help them get used to the cues and make it easier to use the auto-completion tools efficiently.

Eventually, as suggested by the literature, age can be an influential factor in the time required to complete some tasks like pointing and clicking. Yet the pool of participants did not include enough middle aged and older participants. Thus, it was impossible to study the influence of the age on the completion times. Future studies must endeavor to include more age diversity within the participants to explore the influence of aging in the completion times and the success in using auto-completion tools.

Conclusion

This research contributes to the advances in the mouse-tracking technology and its applications to improve the usability of laptops. Indeed, I created a mouse-tracking device that records the cursor position on the screen and computes the velocities in real time. The real time computations are paving the way for auto-completion tools. The first application I presented demonstrates how we can look for different patterns at once to facilitate the navigation on a web browser by defining points of interest and automatically moves the cursor to these points when the movement of the cursor points to a given target. The second application shows how we can predefine a pattern that once recognized by the mouse-tracking device, triggers a tool that assists the user in text selection. Finally, the last tool helps scrolling a page without moving the cursor on the small icons of the scrollbar.

The results obtained from the experiment carried out to validate the usability of these tools gave several insights. First, regarding the navigation improvement tool, it showed that participants found this tool useful and the time required to complete these tasks was reduced for the participants from the experimental group. Using Fitts' law indicated that movements' time for the navigation tasks were lower than expected and, in particular, participants from the experimental group were almost twice as fast as the prediction from Fitts' law.

As anticipated, the cue for the text selection assistance tool was hard to draw precisely and this tool could not be used efficiently on the first attempt as the significantly higher completion time proved. Finally, the scrolling tool was characterized by a very high usability score indicating that the participants found it easy to draw the cue and that

the tool was helpful even if the completion time indicated that experimental group needed more time. These results tend to show that participants were satisfied with a solution that makes it easier to use a laptop's touchpad even if they need some time to get used to it.

This study also proved that analyzing mouse-tracking data can help implementing auto-completion tools that detect specific cues or movements toward specific target points but to be efficient and useful, the patterns need to be intuitive and easy to draw. Several areas of improvements of the current tools were identified and we hope that this will lead to more research in this field.

References

1. Armbrüster, C., Sutter, C., & Ziefle, M. (2007). Notebook input devices put to the age test: the usability of trackpoint and touchpad for middle-aged adults. *Ergonomics*, *50*(3), 426-445.

2. Bevan, N. (2006). Practical issues in usability measurement. *Interactions*, *13*(6), 42-43.

3. Boritz, J., & Cowan, W. B. (1991). Fitts's law studies of directional mouse movement. *human performance*, *1*(6).

4. Chen, M. C., Anderson, J. R., & Sohn, M. H. (2001, March). What can a mouse cursor tell us more?: correlation of eye/mouse movements on web browsing. In *CHI'01 extended abstracts on Human factors in computing systems* (pp. 281-282). ACM.

5. Cooke, L. (2006, May). Is the Mouse a" Poor Man's Eye Tracker"?. In *Annual Conference-Society for Technical Communication* (Vol. 53, p. 252).

6. Gajos, K. Z., Wobbrock, J. O., & Weld, D. S. (2008, April). Improving the performance of motor-impaired users with automatically-generated, ability-based interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (pp. 1257-1266). ACM.

7. Guo, Q., & Agichtein, E. (2008, July). Exploring mouse movements for inferring query intent. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 707-708). ACM.

8. Guo, Q., & Agichtein, E. (2010, April). Towards predicting web searcher gaze position from mouse movements. In *CHI'10 Extended Abstracts on Human Factors in Computing Systems* (pp. 3601-3606). ACM.

9. Holmqvist, K., & Wartenberg, C. (2005). The role of local design factors for newspaper reading behaviour-an eye-tracking perspective. Lund University Cognitive Studies, 127, 1-21.

10. Holzinger, A., Searle, G., Kleinberger, T., Seffah, A., & Javahery, H. (2008, July). Investigating usability metrics for the design and development of applications for

the elderly. In *International Conference on Computers for Handicapped Persons* (pp. 98-105). Springer, Berlin, Heidelberg.

11. Huang, J., White, R., & Buscher, G. (2012, May). User see, user point: gaze and cursor alignment in web search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1341-1350). ACM.

12. Huang, J., White, R. W., & Dumais, S. (2011, May). No clicks, no problem: using cursor movements to understand and improve search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1225-1234). ACM.

13. Kang, Z. and Bass, E.J. (2014). Supporting the eye tracking analysis of multiple moving targets: Design concept and algorithm. *IEEE International Conference on Systems, Man, and Cybernetics*. San Diego, CA, pp. 3191-3196.

14. MacKenzie, I. S. (1992, September). Movement time prediction in human-computer interfaces. In *Proceedings of Graphics Interface* (Vol. 92, No. 7, p. 1).

15. Mueller, F., & Lockerd, A. (2001, March). Cheese: tracking mouse movement activity on websites, a tool for user modeling. In *CHI'01 extended abstracts on Human factors in computing systems* (pp. 279-280). ACM.

16. Murata, A. (1998). Improvement of pointing time by predicting targets in pointing with a PC mouse. *International Journal of Human-Computer Interaction*, *10*(1), 23-32.

17. Oel, P., Schmidt, P., & Schmitt, A. (2001, September). Time prediction of mouse-based cursor movements. In *Proceedings of Joint AFIHM-BCS Conference on Human-Computer Interaction IHM-HCI* (Vol. 2, pp. 37-40).

18. Pusara, M., & Brodley, C. E. (2004, October). User re-authentication via mouse movements. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security* (pp. 1-8). ACM.

19. Quétard, B., Quinton, J. C., Mermillod, M., Barca, L., Pezzulo, G., Colomb, M., & Izaute, M. (2016). Differential effects of visual uncertainty and contextual guidance on perceptual decisions: Evidence from eye and mouse tracking in visual search. *Journal of Vision*, *16*(11).

20. Räihä, K. J., Aula, A., Majaranta, P., Rantala, H., & Koivunen, K. (2005). Static visualization of temporal eye-tracking data. Human-Computer Interaction-INTERACT 2005, 946-949.

21. Rodden, K., Fu, X., Aula, A., & Spiro, I. (2008, April). Eye-mouse coordination patterns on web search results pages. In *CHI'08 extended abstracts on Human factors in computing systems* (pp. 2997-3002). ACM.

22. Sauro, J., & Kindlund, E. (2005, April). A method to standardize usability metrics into a single score. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 401-409). ACM.

23. Smeding, A., Quinton, J. C., Lauer, K., Barca, L., & Pezzulo, G. (2016). Tracking and simulating dynamics of implicit stereotypes: A situated social cognition perspective.

24. Smith, M. W., Sharit, J., & Czaja, S. J. (1999). Aging, motor control, and the performance of computer mouse tasks. *Human factors*, *41*(3), 389-396.

25. Trewin, S., Keates, S., & Moffatt, K. (2006, October). Developing steady clicks: a method of cursor assistance for people with motor impairments. In *Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility* (pp. 26-33). ACM.

26. Vitu, F., O'Regan, J. K., Inhoff, A. W., & Topolski, R. (1995). Mindless reading: Eye-movement characteristics are similar in scanning letter strings and reading texts. Attention, Perception, & Psychophysics, 57(3), 352-364

27. Wilson, K. A., Heinselman, P. L., and Kang, Z. (2016). Exploring applications of eyetracking in operational meteorology research. *Bulletin of the American Meteorological Society*, 97(11), 2019-2025.

28. Ziebart, B., Dey, A., & Bagnell, J. A. (2012, February). Probabilistic pointing target prediction via inverse optimal control. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces* (pp. 1-10). ACM.

Appendix A: Usability questionnaire

Participant's ID:

Participant's Age:

I had access to the auto-completion tool: Yes ☐ No ☐ (If No, answer only the first question below)

In general, I find it hard to use the touchpad: Yes ☐ No ☐

All the items use a five point scale from strongly disagree to strongly agree:

	Strongly Disagree 1	2	3	4	Strongly Agree 5
1. I found the tasks hard to perform	☐	☐	☐	☐	☐
2. It is easy to understand how auto-completion tools work	☐	☐	☐	☐	☐
For the following tasks, evaluate how useful was the auto-completion tool:					
3. Moving the cursor to the URL address bar	☐	☐	☐	☐	☐
4. Moving the cursor to the search bar	☐	☐	☐	☐	☐
5. Moving the cursor to the Google menu	☐	☐	☐	☐	☐
6. Select text for further modification	☐	☐	☐	☐	☐
7. Scroll down/up a web page	☐	☐	☐	☐	☐
8. Overall, I found these tools useful	☐	☐	☐	☐	☐
9. The navigation was harder due to the tools	☐	☐	☐	☐	☐