A FRAMEWORK FOR GAZE REGION ESTIMATION ON WEB PAGES

JOHN KINYUA NJUE

MASTERS OF SCIENCE

(Software Engineering)

JOMO KENYATTA UNIVERSITY OF

AGRICULTURE AND TECHNOLOGY

2019

A Framework for Gaze Region Estimation on Web Pages

John Kiyua Njue

A Thesis submitted in partial fulfillment for the Degree of Masters of science in Software Engineering in the Jomo Kenyatta University of Agriculture and Technology

2019

DECLARATION

This thesis is my original work and has not been submitted for a degree in any other university.

Signature: …… Date: ……

John Kiyua Njue

This thesis has been submitted for examination with my approval as the university supervisor.

Signature: …… Date: ……

Prof. Waweru Mwangi, PhD

JKUAT, Kenya

Signature: …… Date: ……

Dr. Michael Kimwele, PhD

JKUAT, Kenya

DEDICATION

This thesis is dedicated to my beloved parents, who wholeheartedly have been very supportive when I thought of giving up. You have always provided the support needed for successful completion.

My supervisors, you did a good job to the last minute ensuring that a quality standards were maintained. Many thanks.

To my wife, I have no words, you have always been there with your encouraging words. To my sisters, uncle, relatives, mentors and friends, you have always been there with words of encouragement to finish this study. My late uncle, you dreamt big about my academic success. R.I.P.

And lastly, I dedicated this book to the Almighty God, thank you for the guidance, strength, power of mind, protection, skills and for healthy life.

You always answer prayers when I turn to you.

ACKNOWLEDGEMENT

I would like recognize and appreciate the School of Computing and Information Technology (SCIT) at Jomo Kenyatta University of Agriculture and Technology (JKUAT) for its generous support to the far I am as a student.

I am also humbled to give my special appreciations to my supervisors; Prof. Waweru Mwangi and Dr. Kimwele for their guidance and constant contribution that made it possible for me to successfully complete this proposal. My warm and special appreciation also goes to my friends and family members for their generous support.

Your sacrifices and support for me have touched my heart. Thank you and May God bless you abundantly.

TABLE OF CONTENTS

DECLARATION..... **ii**

DEDICATION..... **iii**

ACKNOWLEDGEMENT..... **iv**

TABLE OF CONTENTS..... **v**

LIST OF TABLES..... **ix**

LIST OF FIGURES..... **x**

LIST OF APPENDICES..... **xii**

LIST OF ABBREVIATIONS AND ACRONYMS..... **xiii**

ABSTRACT..... **xiv**

CHAPTER ONE..... **1**

INTRODUCTION..... **1**

1.1 Background..... 1

1.2 Problem Statement..... 3

1.3 Objectives..... 3

1.3.1 General Objective..... 3

1.3.2 Specific Objectives..... 3

1.4 Research Questions..... 4

1.5 Justification..... 4

1.6 Scope of study..... 6

CHAPTER TWO..... 7

LITERATURE REVIEW..... 7

2.1 Introduction..... 7

2.2 Human Gaze, Focus and Attention..... 9

2.2.1 Gaze..... 9

2.2.2 Attention and Gaze Focus..... 10

2.3 Eye/Gaze Tracking..... 11

2.3.1 Gaze Tracking Methods and Techniques..... 12

2.3.2 Eye Tracking Metrics..... 13

2.3.3 Gaze Fixation..... 14

2.3.4 Fixation Identification Algorithms..... 15

2.4 User Activities and Behaviors on the Web..... 18

2.4.1 Tracking User Behaviors and Activities..... 20

2.4.3 Tracking Users Activities in Web Interfaces..... 22

2.5 Mouse Activities, Keyboard Activities and Document Scrolls and Their Effects on Gaze Region..... 25

2.5.1 Mouse activities and Gaze Tracking..... 25

2.5.2 Keyboard activities and gaze region..... 31

2.5.3 User document Scrolls activities and gaze region..... 34

2.5.4 Eye/Gaze region Estimation Methods and Techniques..... 36

2.7 The Conceptual Framework..... 41

2.7.1 Determining gaze position from mouse activities..... 42

2.7.2 Determining Gaze position from document scroll activities..... 43

2.8.3 Determining Gaze position from keyboard activities..... 44

CHAPTER THREE..... 46

METHODOLOGY..... 46

3.1 Introduction..... 46

3.2 Research Design..... 47

3.3 Population..... 48

3.4 Population Sampling..... 49

3.5 Data Collection Procedure..... 49

3.6 Data Analysis..... 50

3.6.1 Classifying data into mouse activities, keyboard activities and scroll activities..... 52

3.6.2 Estimating Gaze with Mouse, Keyboard and Scroll Interactions..... 54

CHAPTER FOUR..... 58

RESULTS AND DISCUSSION..... 58

4.1 Introduction..... 58

4.2 Descriptive Analysis..... 59

4.2.1 Percentage comparison of activities and time spent using mouse, keyboard and scroll..... 59

4.2 Gaze estimation..... 62

4.3 Path followed and gaze estimated from path followed when using mouse, keyboard and scroll per session..... 63

4.4 Results Validation.....70

CHAPTER FIVE..... 73

SUMMARY AND CONCLUSION..... 73

5.1 Summary.....73

5.2 Conclusion..... 74

5.3 Recommendations.....75

REFERENCES..... 76

APPENDICES.....85

LIST OF TABLES

Table 3.1: Data collected via browser script.....51

LIST OF FIGURES

Figure 2.1: Eye fixations separated by saccades..... 16

Figure 2.2: An example clustering area with three clusters.....17

Figure 2.3: Mouse trails recorded by mouse tracker..... 28

Figure 2.4: The ClickTale generated heat maps.....29

Figure 2.5: Heat maps of click positions recorded cursor positions.....30

Figure 2.6: EyePoint technique for pointing and selection..... 34

Figure 2.7: Conceptual Framework.....42

Figure 2.8: Mouse click at position X1, Y1 considers X1, Y1 as gaze position at time t_i.....43

Figure 2.9: Mouse cursor moving from X_1, Y_1 at time t1 to X_2, Y_2 at time t_2 will consider X_2, Y_2 at current gaze fixation at time t_2.....43

Figure 2.10: Document scrolling vertically.....44

Figure 2.11: Cursor Insertion Point.....44

Figure 2.12: Using keyboard to input text.....45

Figure 2.13: Using keyboard navigation keys.....45

Figure 3.1: Client Server Architecture..... 48

Figure 3.2: Classifying data into mouse activities, keyboard activities and scroll activities.....53

Figure 3.3: Algorithm for classifying data into mouse, keyboard and scroll activities 57

Figure 4.1: Comparison of events captured from mouse, keyboard and scroll..... 59

Figure 4.2: Comparison of time spent using mouse, keyboard and scroll.....60

Figure 4.3: Comparison of Activities Count for mouse, keyboard and scroll.....61

Figure 4.4: Comparison of time spent on event for mouse, keyboard and scroll..... 62

Figure 4.5: Points of focus taken from the scroll activities data..... 63

Figure 4.6: Gaze path estimated using scroll activities data.....65

Figure 4.7: Points of focus taken from the keyboard activities..... 66

Figure 4.8: Gaze path estimated using keyboard activities data.....67

Figure 4.9: Points of focus taken from the mouse activities..... 68

Figure 4.10: Gaze path estimated using mouse activities data..... 69

Figure 4.11: Aggregated gaze path.....71

Figure 4.12: Activities Timeline.....72

LIST OF APPENDICES

Appendix I: Table of sessions randomly selected for analysis..... 85

Appendix II: Table of Data for session with id 1459517689455.....87

Appendix III: Sample Codes for Gaze Classification.....88

Appendix IV: Sample Codes for Gaze Estimation.....89

Appendix V: Sample Javascript Codes for Data Collection.....94

Appendix VI: Sample Question and Answer Questions used in Data Collection.....99

LIST OF ABBREVIATIONS AND ACRONYMS

ROI	Region of Interest
ACE	Adaptive Coach for Exploration
AJAX	Asynchronous JavaScript and XML
ANN	Artificial Neural Network
AOI	Areas of Interest
API	Application Programming Interface
DOM	Document Object Model
HCI	Human Computer Interaction
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ILE	Intelligent Learning Environment
IUI	Intelligent User Interfaces
SQL	Structured Query Language
POR	Points of Regard
REST	Representational State Transfer
RNNs	Regression Neural Networks
SERPs	Search Engine Results Pages
SVM	Support Vector Machine
WIMP	Window, icon, menu and pointer
WWW	World Wide Web

ABSTRACT

Gaze estimation and tracking systems largely depend on head-mounted cameras which make them cumbersome to use, thus, they are restricted to laboratory set up. In recent studies, researchers have attempted to estimate gaze using mouse cursor which has produced inaccurate results because factors that influence eye-gaze region have not been considered. This thesis examined existing methods for gaze estimation then proposed a framework for estimating gaze region using enhanced cursor based trackers. The work investigated mouse activities, keyboard activities and document scrolls activities on web pages which together were used to estimate gaze region in web page documents. The main tool used this study for data collection was a web application with client-side code for recording mouse, keyboard activities and document scrolls while users navigate through web page documents. Data collected was pushed to remote web server for storage and analysis. Analysis involved two stages. Stage one involved extraction of scroll activities from mouse and keyboard activities because scroll can be achieved by either of these. Stage two involved use of Lindeman, Merenda, and Gold (LMG) metric linear regression model to estimate gaze from mouse, keyboard and scroll data. The main outcome of this work was a framework for estimating users gaze. To achieve this, we first came up with a theoretical model describing relationship between mouse activities, keyboard activities, document scrolls activities and the user gaze. An algorithm that uses LMG metric linear regression model was then used to estimate users region from collection mouse, keyboard and scroll activities data. The results shows that mouse activities only cannot be relied on to estimate user's gaze. Both keyboard and scroll activities are play a significant role in gaze over a period of time and must therefore be considered when estimating user's gaze region.

Index Terms: Gaze tracking, gaze region, linear regression model, mouse activities, keyboard activities, scroll activities, web page.

CHAPTER ONE

INTRODUCTION

1.1 Background

Identifying user attention or concentration of the mind on web pages has become an area of interest for many researchers in their attempt to study user interests. This is because the World Wide Web is one of the primary means of acquiring information and communication throughout the world. Millions of people visit the Internet using their browsers as part of their daily life. Individuals therefore navigate from one web page to another in an attempt to locate information that is relevant to their interest until they find a source of information that catches their attention before posing to read such information. In search engine results pages (SERPs), understanding how people interact with search engines is important in terms of improving search quality and therefore making it easier to locate information (Huang, White & Dumais, 2011).

One of the main ways of studying attention on web pages is by tracking a user's eye gaze as well as eye movements. These aspects are important as they provide rich details on user attention by sampling eye gaze position at different time intervals. In this regard, eye gaze tracking records eye movements while a participant examines a visual stimulus on a particular webpage (Mansour & Flowe, 2010).

Steichen *et al.* (2013) identified that eye tracking has been used in cognitive and perceptual psychology to analyze user attention patterns in information processing tasks. Moreover, within the realm of human-computer interaction and information visualization, eye-tracking technology is being employed to investigate trends and differences in user attention patterns and cognitive processing. For instance, this technology is used to study different interfaces/visualization types, task types such as reading and mathematical reasoning, or user abilities such as cognitive abilities and expertise.

During eye tracking, one acquires and analyzes the eye movement data, which can then be used to determine where user's attention is focused. One is then able to determine gaze direction of a person at given time and also the sequence in which the eyes move (Anacan *et al.*., 2013). Huang *et al.* (2012) identified that by capturing and analyzing detailed eye movements of a user reading a web page can reveal more concerning the ways in which interaction with the page occurs. By watching people as they use web pages for specific tasks, it is possible to reveal their interests easily if eye gaze concentration areas and movements are captured and tracked. Various approaches have been used to track the eyes and can be grouped into three: contact lens based, electrocardiogram based and video based eye trackers (Chennamma & Yuan, 2013).

Recent studies have in this respect considered a user's mouse activity as a source of approximate eye gaze position while the user looks at the web page with some degree of accuracy. In the studies, mouse data was collected by tracking mouse cursor position (Ooms *et al.* 2015).

Huang *et al.* (2012) observed that cursor behaviors include inactive, examining, reading and action. Their work concluded that adequate cursor data can be used more effectively as an alternative to traditional eye trackers. Other studies, have concluded that eye gaze follows mouse cursor position to some degree and reasonable correlations between the eye and mouse have been identified. This has led to conclusion that mouse can be used as an alternative eye tracker but the accuracy of gaze region at a particular time is not guaranteed. In regards to web documents, eye gaze tracking is possible because the browser can easily reveal information relating to mouse and document movements. Understanding how people interact with web pages can aid in devising ways to improve interactions on the web. It is for this reason that usability professionals and site designers seek to optimize the experience by analyzing which parts of the page grab a visitor's attention and the information users read to enhance and enhance the experience of navigation.

The results of eye movement studies have been used to understand processes like reading, to infer user intentions and to diagnose medical conditions (Huang *et al.*., 2011).

1.2 Problem Statement

Despite attempts by recent studies to use mouse cursor data to track human gaze, they have provided inaccurate gaze region identification results. This is the result of failing to consider other factors that influence eye-gaze. Several factors including, time, mouse behaviors, document scroll patterns among others affect computer user gaze position. Moreover, inaccurate determination of gaze position has been aggravated by advancements in web design as well as development techniques which, despite having greatly improved web interactivity, have made it difficult to track user attention in dynamically loaded content especially when the user changes interaction method from mouse to keyboard keys to navigate through the dynamic content. Consequently, the existing techniques have been rendered inadequate in gaze region identification due to the limited factors considered in their development, Huang *et al.* (2012).

1.3 Objectives

1.3.1 General Objective

The overall objective of this research exercise was to develop a framework that estimates user's gaze on web interfaces from user activities of mouse, keyboard and scroll with the help of linear regression models.

1.3.2 Specific Objectives

i. To investigate mouse activities, keyboard activities and document scrolls and their effect on eye gaze region.

ii. To develop a conceptual framework for the relationship between mouse activities, key board activities, document scrolls and eye gaze region.

iii. To implement a prototype based on the framework.

iv. Evaluation of the developed framework.

1.4 Research Questions

i. What is the relationship among mouse activities, keyboard activities, document scrolls to eye gaze region?

ii. How is eye gaze region identification enhanced by keyboard and scroll activities on web pages?

iii. How is the eye gaze identification tool implemented?

iv. Is gaze estimated using the framework more accurate than gaze estimated using mouse data?

1.5 Justification

Unlike modern web documents that are made up of dynamic web content, images and nonlinear page layouts, which has made information search process an increasingly complex task, traditional web pages consisted of static web pages interlinked by hyperlinks. Conventional search pages were thus simple, text only and contained a linear listing of documents. Today, web pages are composed of interactive elements, images and text in multiple colors, font sizes, and varying indentation. They include multi-column layouts and contain various page elements drawn from news, images, documents, maps and facts. The result of this is complex multiple page elements competing for the user's attention is thus critical. With attention being a limited resource, understanding which page elements get more or less attention is important (Navalpakkam *et al.*., 2013).

Attempts to study attention on web pages has always focused on the linear page layout (containing a single column of information) and has always concluded that users pay most attention to the top-left of the page, and attention decreases as they

move towards the right or bottom of the page. Nonetheless, this is not always the case in a number of instances. To study user attention on the web, researchers use eye tracking which provides rich details on user attention by sampling eye gaze position at different time intervals (Feit *et al.*., 2017)

The main method for determining user gaze position/region is by tracking the eye gaze. Eye/gaze tracking is mainly done using head mounted cameras. This process entails collecting data as a user performs tasks on the user interface as well as identifying areas of interest as inferred from collected data (Cognolato *et al.*., 2018).

An important indicator of an area of interest is gaze position-where a user concentrates attention. Eye/gaze tracking is conventionally done using cameras as it provides accurate estimates of user eye gaze. However, it has a limitation because the studies can only take place in controlled lab settings among other logistical challenges. Research has also identified that wearable and image-based methods can be used to track user focus while navigating an interface. These methods are associated with a high degree of accuracy associated, but they are expensive to due to associated costs, demand for user calibration and the fact that they can only work within a laboratory setup. Such tracking techniques cannot, therefore, be used to track user focus in daily life (Guo & Agichtein, 2010).

Recent research has considered a user's mouse activity as a source of approximate eye gaze position while the user looks at the web page to some degree of accuracy. Data is collected by tracking mouse movement and this includes cursor position. Some research studies have identified that eye gaze follows mouse cursor position to some degree and reasonable correlations between the eye and mouse have been identified. This has led to conclusion that mouse can be used as an alternative eye tracker but the accuracy of gaze region at a particular time is not guaranteed (Al-Rahayfeh & Faezipour, 2013)

Although the research on mouse-based trackers shows a relationship between eye gaze and mouse cursor, factors that affect mouse cursor position and hence gaze region on a document during searching and reading have not been considered.

Current and previous research gives an indication that cursor data can indeed be used to determine user gaze region and path. However, this determination has not been successful due to inaccuracies associated with using mouse cursor data only. This might be caused by not considering factors such as keyboard activities and document scrolls that may influence user gaze. Again, not all aspects of mouse activities have been considered (Huang *et al.,* 2012).

This research, therefore, contributed to cursor-based eye tracking by investigating ways through which accuracy of cursor trackers can be enhanced. This will be realized by considering additional parameters that affect accuracy of gaze region positively and can thus be used to predict the gaze region more accurately. Factors considered include elements of mouse activities, keyboard activities and document scrolls. When considered together, it is possible to determine gaze regions more accurately as opposed to using cursor data only.

It is hoped that the research undertaking will have a wide application in user attention research in large scale as it does not require controlled lab settings, expensive wearable equipment as well as demand for user calibration.

1.6 Scope of study

This research exercise investigated the influence of mouse activities, keyboard activities and document scrolls with a view to identifying gaze regions on web pages.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

The World Wide Web (WWW) or the web is a rapidly evolving to a ubiquitous information and application medium, which has made it possible to access personal, business or any other form of information in a location-independent and timeless manner. This ability has significant impact on how people, businesses and learning among others access information and services. The pioneer to such access is web-based applications that allow audiences with heterogeneous goals, preferences, and capabilities to access the web from a diverse locations and devices. Web-based applications are complex and information-intensive in that they handle huge data records from wide range of sources. A significant feature of these web-information and intensive applications is the non-sequential navigation of information items on web documents. This means that there is no definite order capable of determining a sequence through which a text is read in web documents (Dolog, 2006).

As millennium celebrations faded, migration from desktop applications to web applications began to gain attention and acceptance as web 2.0-a transition from static HTML Web pages to a more dynamic web- became a reality and its popularity hit the ground. Researchers further argue that the number of people, organizations, and institutions relying on web applications to manage their data and resources has increased significantly. Schneider *et al.* (2008)

The recent popularity of asynchronous enabled web sites has also caused a fundamental shift in the classical HTTP request-response of the Web. Widespread implementation of this approach is usually executed through Asynchronous JavaScript and XML (AJAX) - a combo of technologies that enable Web browsers to send and request data from the server asynchronously. The result is the automation of HTTP requests rather than human-generating such requests. In AJAX, web pages

contain embedded request-response functions comprising a JavaScript application engine that automatically executes in the background to asynchronously pre-fetch large quantities of data from the server (Bellettini, Marchetto & Trentini, 2004).

Wide application of AJAX is being used to enhance interactivity and user experience in web applications. Towards this end, Google Maps, Gmail and Yahoo!, Facebook and Twitter are classic examples of AJAX applications. With AJAX content, sections of a web-documents change from time to time, as they get filled with new content. Most AJAX applications consist of a single page whose elements are updated dynamically in response to callbacks activated asynchronously by the user or by a server message (Bellettini, Marchetto & Trentini, 2004).

A shift in web page design has also significantly influenced the manner in which people access information on web as well as web applications (Altaboli & Lin, 2011). This is different from classical web sites and applications that consisted of a linear listing of documents composed of simple, text only static HTML web pages describing the content. Further, the pages were based on a multi-page interface model, in which interactions are based on a page-sequence paradigm and interactions between web pages is mainly achieved through hypertext links so that for every request, the entire interface is refreshed (Darwish & Lakhtaria, 2011).

Modern web pages on the other hand are made up of dynamic web content, images and nonlinear page layouts. The documents include multi-column layouts where page elements are drawn from news, images, documents, maps and facts (Mesbah & Deursen. 2007).

While Classical web applications model are simple and elegant in design in terms of exchanging documents, the model has many limitations for developing modern web applications underlined by friendly human-computer interaction due loss of user interactivity available in traditional desktop applications. The pages of current applications are composed of interactive elements, images and text in multiple colors, font sizes, and varying indentation. This dynamic web content, images and nonlinear page layouts has made information search process an increasingly complex task. The

result of this is complex multiple page elements competing for the user's attention is thus critical. With attention being a limited resource, understanding which page elements get more or less attention is important (Navalpakkam *et al.*, 2013).

2.2 Human Gaze, Focus and Attention

2.2.1 Gaze

Friedman, Langhans and Lee (2016) define gaze as the orientation of the eyes when viewing a particular point and is an important visual cue in communication. Gaze occurs during scene perception when high-quality visual data is recorded only from a limited spatial location encompassing the center of the fovea/of gaze (Zhang, Bulling, & Gellersen, 2013). These researchers further note eyes are move about three times each second through speedy eye movements (saccades) to re-orientate the fovea throughout the scene. Gaze pattern information is only acquired during periods of relative gaze stability (fixations) owing to saccadic suppression during the saccades themselves (Yokoyama *et al.*, 2014).

Since gaze indicates a person’s current line of sight at point of fixation, it can be used to interpret the user’s intention for non-command interactions. The gaze may also provide several functions in communication, such as giving cues of people’s interest and attention, facilitation during conversations, giving reference cues by looking at an object or person, and indicating interpersonal cues such as friendliness or defensiveness (Zhu & Yang, 2015).

Gaze perception plays an important role in everyday lives because it conveys ones desires and intentions. In addition, user's behavior can be predicted by perceiving their gaze, because gaze direction signals the upcoming target or goal. Therefore, gaze perception is a significant cognitive function that facilitates interactions (Yokoyama *et. al.*, 2014). In web navigation, people move their eyes from place to place within the browser view point until person`s eyes are attracted by points of interest on the web site when the person`s attention is captured and the brain draws a picture of the process (ROBU, 2013).

2.2.2 Attention and Gaze Focus

Attention is achieved when an organism is ready to perceive stimuli surrounding it. Through attention, one sustains concentration on a stimulus, sensation, idea, thought or activity. Focus on the other hand occurs when one gives complete attention to an object, idea, endeavor, action. Attention thus plays a central role in visual and cognitive processing and since eye movements are an overt behavioral manifestation of the allocation of attention within a scene, eye movements serve as a window into the operation of the attention system. In this regard, gaze may be focused or non-focused. Gaze perception may also require a focus of attention thus the direction of person's gaze is difficult to ignore when presented at the center of attention (Yokoyama *et al.*, 2014).

Burton *et al.*, (2009) identified that gaze perception requires focused attention. This means that gaze direction cannot be perceived outside the focus of attention. Gaze perception requires a focus of attention in an averted manner (e.g., leftward or rightward) as opposed to a direct manner. As opposed to an averted gaze, a direct gaze, is processed in the brain and as such does not require a focus of attention (Yokoyama *et al.*, 2014).

Henderson (2003) observes that a user can control gaze by directing fixation through a scene in real time in the service of ongoing perceptual, cognitive and behavioral activity. In this regard, Gaze control is important in scene perception because vision is an active process in which the viewer seeks out task-relevant visual information. Active vision ensures that high quality visual information is available when it is needed, and also simplifies a variety of otherwise difficult computational problems.

Eye gaze is a reliable indicator of what a person is thinking when the direction of gaze carries information about the focus of the user’s attention. Gaze focus can be measured using fixation intensity which is an important feature for predicting users’ attention. Thus, examining how eye gaze contributes to identification of a user’s attention of during human-machine conversation (interaction) is important (Prasov, *et al.*, 2007).

A user's eyes do not always remain focused on the task as he/she may look at areas of interest for less than 50% of the task time on average for easy tasks. For difficult tasks, a user can look at the screen 70% of the task time. This suggests that for more difficult tasks, users tend to concentrate more on specific areas of the screen compared to easier tasks (Iqbal & Bailey, 2004). On the same breadth, Shuhong and Qiaorong (2014) observed that in web interfaces, attention shifts from one interesting region to another when location saliency within a particular region decreases and is replaced by the next salient location.

2.3 Eye/Gaze Tracking

Eye/gaze tracking records eye movements while a participant examines a visual stimulus and captures what the viewer is focusing on by gathering data of eye movements' data and the gaze point (Boczon, 2014).

Through eye tracking, one can easily uncover exactly where users look at on the screen and when they use a web page or software application. To achieve this, participants are asked to carry out a task, and then where their eyes are pointing is measured. Eye tracking thus measures the spatial direction where the eyes are pointing and returns information into what observer found interesting and how the observer perceived the scene he was viewing by following the path of an observer's visual attention. The intuitive sense of visual attention from a psychological point of view is the taking possession by the mind, in clear and vivid form of one out of what seen as several simultaneously possible objects or trains of thought (Chabane *et al.*, 2006).

For eye tracking to be accurate, one must account for both the position of the head in addition to the position of the eyes relative to the head. The history of eye trackers is associated with mechanical devices that must have caused users great discomfort due to their size and invasiveness. Using existing technology, one has to wear a headgear or any physical attachments to have his/her eyes tracked (Mansour & Flowe, 2010).

2.3.1 Gaze Tracking Methods and Techniques

Various approaches have been used to track the eyes and can be grouped into three: contact lens-based, electrocardiogram-based and video-based eye trackers. Common eye tracking methods include electrooculography, Limbus, Pupil and Eyelid Tracking, Contact Lens Method, Cornea1 and Pupil Reflection Relationship, Purkinje Image Tracking, Artificial Neural Networks and Head Movement Measurement (Chennamma &Yuan, 2013).

Tracking the data regarding gaze point and eye movements can be achieved through absolute and relative tracking techniques. Absolute eye tracking can be achieved via infrared based eye trackers that use infrared light emitted by devices such as tablet computers. These techniques demand calibration of the eyes. During tracking, infrared sensors are used to detect and register reflections of infrared light projected on the eyes. These reflections are known as corneal reflections. Gaze point is then derived from observing the relation between the reflection of the cornea and the pupil center. Absolute eye tracking aims at providing the actual point of gaze with data being acquired remotely. Relative eye tracking on the other hand is achieved through wearing head mounted eye trackers that measure the orientation and movement of the eyes. Common methods used in relative eye tracking are electrooculography, contact lens, and head-mounted infrared systems (Boczon, 2014).

Eye tracking systems fall into two categories: analytical approaches that compute eye gaze by analyzing locations of certain facial features; and neural network approaches that use eye images as inputs thus relieve user from any head-mounted equipment (Zhu & Yang, 2015). Eye tracking system can further be grouped into wearable or non-wearable, and infrared-based or appearance-based systems. In infrared-based systems, a light shining on the subject whose gaze is to be tracked creates a “red-eye effect” produced by the difference in reflection between the cornea and the pupil. This effect is used to determine the direction of sight. In appearance-based systems, computer vision techniques are used to find the eyes in the image before proceeding to determine their orientation (Mansour & Flowe, 2010).

Gaze tracking methods can also be classified into intrusive techniques and non-intrusive. Intrusive techniques such as measuring the reflection of some light (usually infrared light) that is shone onto the eye, measure the electric potential of the skin around the eyes by applying special contact lenses that facilitate eye gaze tracking. Non-intrusive methods estimate the eye-gaze based on a neural network. Neural networks can accurately estimate the position of the eye gaze on a computer screen by using images observed user's eyes as input. This form of gaze trackers however, require training prior to usage (Stiefelhagen, Yang & Waibel, 1997).

2.3.2 Eye Tracking Metrics

Lupu, Stan and Ungureanu (2014) demonstrated that usability of eye tracking systems can be assessed by metrics relevant to the tasks and their inherent cognitive activities. The most important metrics include:

i. **Fixation**− the time taken for processing image by the fovea;
ii. **Saccade** – the time taken by the fovea to focus its attention from one image to another (time interval between two fixations);
iii. **Gaze Duration**: cumulative duration and average spatial location of a series of consecutive fixations within an area of interest. Gaze duration typically includes several fixations and may also consist of the relatively small amount of time for the short saccades between these fixations;
iv. **Area of interest** − area of display or visual environment that is of interest to the research or design team.
v. **Scan Path**− spatial arrangement of a sequence of fixations.
vi. Number of fixations on each area of interest.
vii. Fixation duration mean on each area of interest.
viii. Total on each area of interest.
ix. Number of areas of interest fixated.
x. Scan path length.
xi. Scan path direction or transition probability between areas of interest.

2.3.3 Gaze Fixation

An important aspect of gaze tracking is eye/gaze fixation. Human eyes movements are characterized by a series of quick jumps or high velocity movements, known as saccades, followed by fixations-periods of time in which the eye is stabilized and remains relatively still.

Cognitive processing of a visual stimulus occurs during fixations while saccades are considered to be voluntary movements of the eye in order to shift focus from one object of interest to another. Identification of the components of eye movements (fixations and saccades) is an essential part in the analysis of visual behavior because these types of movements provide the basic elements used for further investigations of human vision (Blignaut & Beelders, 2009).

Using saccadic eye movements, one is able to accurately orient a fovea within visual field for detailed information acquisition. These movements can result in saccadic adaptation if post-saccadic visual error are consistent over a period of time and when multiple saccades occur (Garaas *et al.*, 2008).

Blignaut and Beelders (2009) further present that during fixations, the eye is subject to different types of low velocity movements, namely tremors; drifts and micro saccades. Tremor is a periodic, wave-like motion of the eyes with a typical frequency of ~90 Hz and amplitude of 20". Fixation hence plays a vital role in the analysis of eye tracking data by allowing the analyst to determine where the subject was looking at any given point. Because of the continuous low-velocity eye movements during fixations, they are described in terms of the mean of the x-y coordinate of the gaze position when measured over a minimum period of time during which the gaze does not move further than a predefined maximum distance.

Although eye-gaze fixation is a positive signal of interest when the user pays more attention to a position, (Huang *et al.*, 2012) identified that prolonged cursor fixation may not be necessarily a gaze since user's attention is probably elsewhere.

2.3.4 Fixation Identification Algorithms

Fixation data can be extracted using existing algorithms. Fixation algorithms can be used to identify fixations within raw gaze data. They are in this respect categorized in terms of the way in which the above-mentioned conditions for fixations and the corresponding thresholds are handled. Fixation extraction algorithms include:

2.3.4.1 Velocity-threshold Algorithm

The velocity-threshold algorithm separates fixation points (Points of Regard (POR) belonging to a fixation) and saccadic points (Points of Regard (POR) that do not belong to a fixation) based on their point-to-point velocities. The velocity of a fixation point is less than a chosen threshold value while a saccadic point has a velocity that is larger than or equal to the threshold. Thereafter, consecutive fixation points are collapsed into fixations and saccadic points are discarded (Blignaut & Beelders, 2009).

2.3.4.2 Dispersion-threshold Algorithm

The dispersion-threshold algorithm utilizes the fact that fixation points, because of their low velocity, tend to cluster close together. Fixations are identified as groups of consecutive points of regard within a particular dispersion or maximum separation. Various metrics can be used for dispersion such as the distance between points in the fixation that are the furthest apart, the distance between any two consecutive points and the distance between points and the center of the fixation. The algorithm suggests that during exploration of the scene or image, the eye has a tendency to remain fixated for a few milliseconds on the most significant areas of an image; after which, the eye moves towards a new zone of interest as indicated in figure Figure 2.1. These significant areas are known as Areas of Interest (AOI), Region of Interest (ROI) or the Points of Regard (POR) included in fixations (Blignaut & Beelders, 2009).

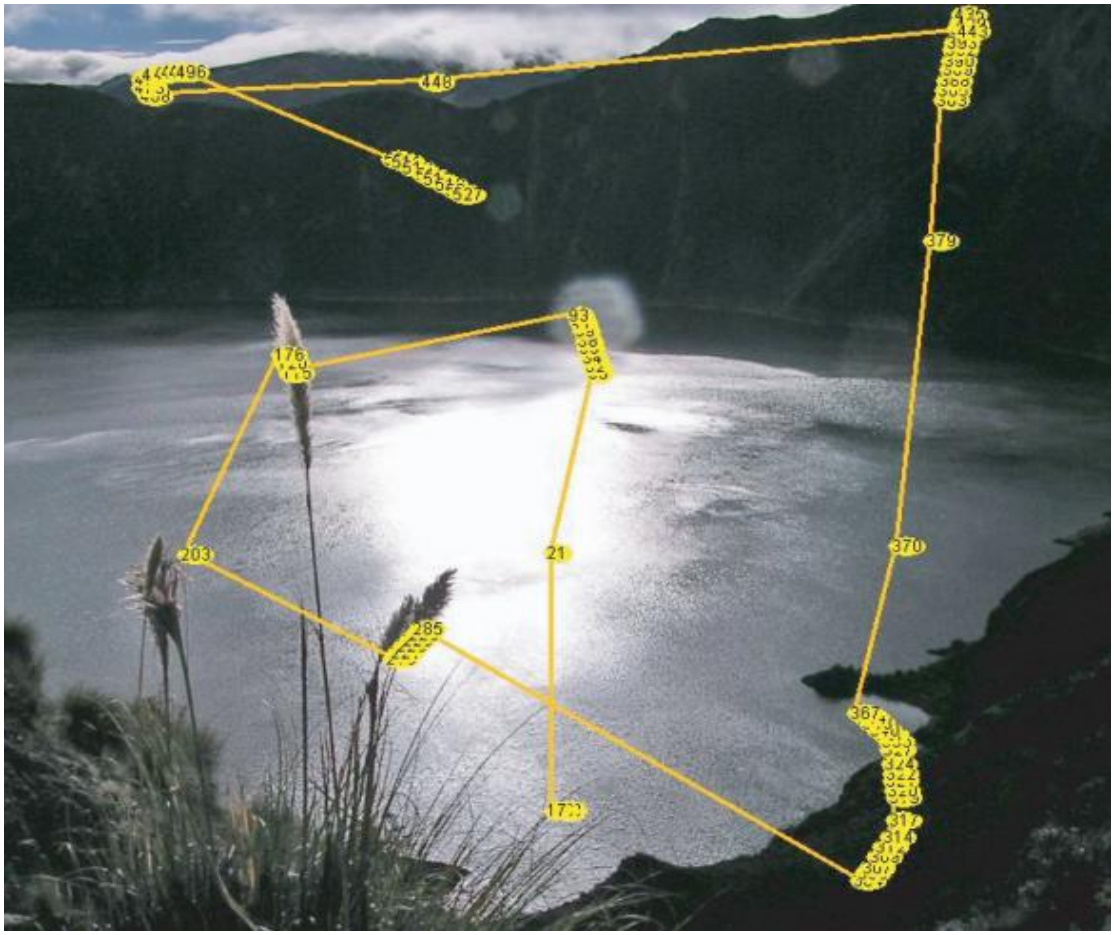

Figure 2.1: Eye fixations separated by saccades (Blignaut & Beelders, 2009)

The spatial dispersion of PORs within fixations depends on the threshold setting for dispersion-based fixation detection algorithm. Other approaches used for fixation detection include:

2.3.4.3 Mean Shift Procedure

The mean shift procedure proposes searches for a local maximum in a d-dimensional space by shifting each point of the space towards higher density areas in order to separate clusters until such movements involve a small number of points (Blignaut & Beelders, 2009).

2.3.4.4 The Clustering Algorithm

The clustering algorithm informs clusters spaces of lower dimension which are then used to identify clusters in the original data set. Projected clustering has mainly been motivated by seminal research showing that, as the dimensional increases, the farthest neighbor of a point is expected to be almost as close as its nearest neighbor for a wide range of data distributions and distance functions. See Figure 2.2. Projected clustering assumes that meaningful structure can be detected only when data is projected onto sub-spaces of lower dimensions.

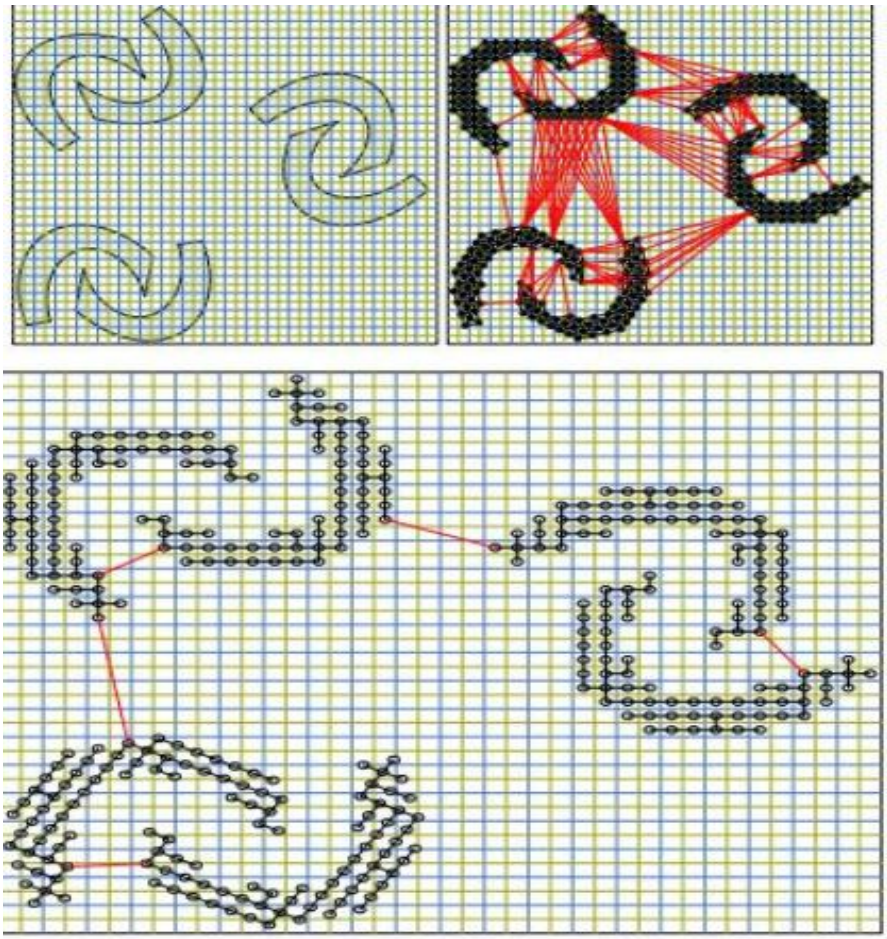

Figure 2.2: An example clustering area with three clusters (Blignaut & Beelders, 2009)

In eye-gaze tracking, this technique clusters eye positions using projections and projection aggregation. The method efficiently deals with eye fixations as multidimensional information. However, the accuracy of fixation identification algorithms is heavily dependent on the parameters specified for minimum duration or maximum area. Thus the resulting fixation point produced by different algorithms and parameter settings will always vary. This may in turn lead to different as well as varying interpretations and conclusions being drawn from the same data depending on the chosen parameter settings.

In the above example, for a given population threshold value, and diameter threshold value, it is possible to generate a flexible grid and thereby obtain a relevant graph. From this, one can generate a minimum spanning tree that makes it easier to get the final clustering. For a fixation to occur in this regard, two basic conditions MUST exist for a cluster of PORs- the total duration must be long enough and the PORs must be close enough (spatially) to one another while forming a temporal sequence (Blignaut & Beelders, 2009).

2.4 User Activities and Behaviors on the Web

Millions of people navigate the Web in search of information by looking, finding, and reading, pointing, and clicking. This prompted website site designers and usability professionals to attempt to optimize the web users' experience by analyzing parts of the pages that grab visitor's attention, as well as the information people read pages (Huang *et al.*., 2012). While people navigate and search through documents and web results, they search for useful information that matches their interests by sorting through large sets of documents to identify useful materials (Dolog, 2008).

Users' interests may be inferred from the activities they undertake while reading and how they interact with individual documents. The interests can be used as a basis for identifying other documents or document elements of potential interest within the set. To effectively identify related documents of interest, people's activity data must be collected from all applications used within a document triage (Dolog, 2008).

Dolog (2008) further notes that user interests can be either implicit or explicit both of which can be identified using systems developed to recognize user interest. Explicit indicators also known as ratings tell the recognizing system how interesting or uninteresting a given document is. Implicit interest indicators record the user's actions instead of relying on users to rate their experiences. These indicators are not obtained directly from user. Instead, one must infer user actions (Schneider *et al.*., 2008).

During web search and navigation, the viewer's eye movement change over time and the change in direction and magnitude are influenced by the type of web sites. The individual characteristics of the viewer and the stimuli contribute to viewers' eye movement behavior. By understanding how web users view different web pages, interaction page order and site type, one can deduce a user's online ocular behavior and the effect on web viewing behavior (Atterer, Wnuk & Schmidt, 2006).

In search engine result pages (SERP), user behavior for different types of tasks differs in various aspects such as search activeness, browsing style, clicking strategy, and query reformulation. Users may shift their interests to focus less on the top results but more on results ranked at lower positions in browsing. With time, the results become less and less attractive to the users. User behaviors may therefore be categorized into three main areas: web search user behavior with eye-tracking data; search task and its effects to user behavior; and search session (Jiang, He & Allan, 2015).

Researchers have attempted to understand the activities and tasks users engage in on the Web by collecting and storing activities using web tracking technologies before connecting these activities to user web browsing behavior. User web activity can be identified by grouping user actions into activities such as reading bulletin board messages, finding product information, or searching for technical support (Atterer, Wnuk & Schmidt, 2006).

2.4.1 Tracking User Behaviors and Activities

Although tracking user eyes remain one of the main and most relevant method of tracking user behaviors and activities, hardware as well as software solutions have also been developed to monitor a vast array of user activities while operating a computing device. These solutions can be obtained at a price or for free by anyone willing to monitor computer usage. Majority of existing solutions monitor users by logging keystrokes typed, application and website usage, detailed file usage, incoming and outgoing chats and e-mails, internet connections, windows interacted with, internet packet data, desktop screen shots, software installations, and more. If a software solution is adopted, all user activities are logged followed by generation of a graphical report. Alerts can also be set for notification when an employee performs some action. (Kaupins & Minch, 2006)

Organizations may also install hardware devices, server computers, and firewalls to track all electronic transactions. Sometimes tracking software may be installed remotely so that they are invisible to the computer user. The technology employed to monitor an employee's activities is thus capable of exposing any action taken by a user on a computer. Towards this end, tracking user behaviors can be achieved through three main approaches: construction of personalized information systems, unobtrusively observing user behaviors and use of web tracking technologies (Yerby, 2013).

2.4.1.1 Personalized information systems

Personalized Information Systems are computer programs/applications whose behaviors are adjusted to specific needs of their users according to the preferences set (Reis & Carvalho, 2012). Construction of personalized information systems can be used to track user behaviors over a period of time. The collated information can then be processed and interpreted to build up a good user model for the required task. By silently observing the behavior of user interaction within an information space, one can gather information such as contexts of an activity, speed and time of an activity, mouse and/or keyboard activity, page navigation information, bookmark selection

information and other user activities on a page using a model (Villa & Chalmers, 2001).

Detailed user activities can be captured by tracking all interaction with the displayed browser page such as moving the mouse pointer around or scrolling the page. During tracking however, interaction should be tracked at the widget level, i.e. mouse coordinates should be mapped to DOM elements like buttons and links. This should be combined with information about the layout of the HTML pages to allow complete tracking of all user activity (Atterer, Wnuk & Schmidt, 2006).

2.4.1.2 Web tracking technologies

User behaviors can also be tracked by capturing information regarding user activities with the help of common web tracking technologies available today. These include:

Clickstreams analysis: involves recording and evaluation of the actions performed by a user on a web site. Data collected includes where visitors enter the web application which is used to identify user goals and predict user actions. Clickstream data can also be collected from web server logs, client-side JavaScript tracking code in the browser or from Internet service providers.

Client identification using Hypertext Transfer Protocol (HTTP) to identify a user over many subsequent web page requests using session information through a process referred to as session handling. This can be achieved through the use of a client's computer IP address; HTTP cookies; web storage; Silverlight isolated storage; Google gears; hidden form fields; query strings; http authentication as well as window.name DOM property.

User tracking techniques such as deep packet inspection, HTTP referrer field in the HTTP header, Web bugs and tracking cookies, Zombie cookies, and Browser Fingerprinting.

Use of privacy enhancing technologies that protect individuals from being tracked and identified on the internet. These technologies involve hiding IP addresses, End-

to-end encryption, Browser-based blocking and plugins, Private Browsing Modes, Opt-out cookies and HTTP Do not track header (Schmucker, 2011).

Information about user activities on the web collected through this approaches has been applied in a wide range of scenarios including Web analytics. This entails measuring and interpreting web site usage data by gathering information like the number of visitors over time, how visitors find out about the web site, effectiveness of marketing campaigns, geographical location of visitors among others (Booth, 2008).

Further, during usability tests for web applications evaluation, JavaScript can be used to capture detailed records of user mouse and keyboard input on web interfaces. Records collected aid in decision making during design and development process (Atterer *et al.*. 2006).

These methods are particularly suited for internet marketing and advertising where users visiting a web site are shown advertisement related to their interests based on search keywords in order to increase the chances of ads catching the user's interest. Behavioral advertising in this regard tries to estimate appropriate advertisements based on a user's profiles based on information like sex, age group, location, estimated income and interests. The information gained from user activities is thus of interest to different parties including advertisement companies, law enforcement and intelligence agencies, usability tests of web applications and web analytics (Schmucker, 2011).

2.4.3 Tracking Users Activities in Web Interfaces

Tracking of users' online behavior and activities is possible because the browser can easily reveal information related to mouse and document movements. Website operators/owners often use information about users' online interactions with their websites by collecting information relating to such aspects as user's identity, display and/or language preference, web pages visited, items purchased, and transactions performed. The purposes of collecting such information vary and may include but

not limited to remembering a user's preference (e.g on language, font size, colour scheme) so that the look as well as feel of a website is recorded for purposes user of subsequent visits; analyzing how users navigate a website with a view to optimizing its design; establishing and maintaining a user's logged-on identity so that he/she can move around the website without being asked to log on again; and tracking the behavior and preferences of an online user with a view to building detailed profiles of the user for serving marketing information or advertisements to suit particular user online behavior (Commisioner, 2014).

2.4.3.1 Unobtrusively observing user behaviors

Goecks and Shavlik (2015) alludes that one can also learn a users' behavior and interests unobtrusively by observing their normal behavior and measuring activities such as the number of hyperlinks clicked on a page or the amount of scrolling performed. User tracking on the web can achieved through collecting and analyzing Search Engine logs that record user actions. Search engines keep records of user interaction with search pages in click-through logs. This happens through the recording temporary user id in logins or cookies, user queries, search engine results and the resulting user clicks on result pages. Data collected through this approach provides implicit information about user web search interests and can also aid in web result pre-fetching, automatic search query reformulation, and click spam detection, estimation of document relevance and prediction of user satisfaction. Moreover, detailed logs about user search activities can provide information on what the user does after having clicked on a document and before he resumes interactions with the search engine (Piwowarski, 2009).

Studying users' search behavior on the web can provide guideline to design and evaluation systems. Earlier studies showed that user behaviors help tailor a system for certain tasks. Users' behaviors therefore vary in terms of activities and change significantly over time (Jiang & Allan, 2015).

2.4.3.2 Use of eye gaze trackers

In web documents, user activities can also be identified by tracking the eye gaze of users as they navigate a website using eye gaze tracking equipment in laboratory setup. The results of lab tests have indicated that one can determine what people are reading by identifying their fixation.

2.4.3.3 Tracking user's mouse activity

Studies have considered user's mouse activity as a source of approximate eye gaze position while the user looks at the web page with some degree of accuracy. Mouse data in this respect is collected by tracking mouse cursor position (Huang *et al.*., 2012).

From the method, understanding how people interact with web pages can help in devising ways to improve interactions on the web. Towards this end, usability professionals and site designers seek to optimize the experience by analyzing which parts of the page grab a visitor's attention and the information users read to enhance and optimize the experience of navigation (Huang *et al.*., 2011).

2.4.3.4. Tracking Users Clicks and Submits

Web site owners can track visitors' clicks on links and form submits in order to assess user activities and improve the web site. Most sites use web beacons to track user actions, but waiting for the beacon to return on clicks and submits slows the next action (e.g., showing search results or the destination page). Typical tracking mechanisms in this regard employ JavaScript to capture the click or form-submit event. The event is suspended as a request is made to a logging server in order to record the user's action before the action is taken. This kind of tracking is achieved through the following mechanisms: (a) Fixed-time. Given a parameter t in msec, web beacon requests are initiated in parallel, and the browser spins for t msec before continuing to the destination independent of whether the tracking requests came back. (b) Out-of-band. Given a parameter **t** in **msec**, the beacon requests are initiated in

parallel, and the browser waits for all of them to come back (maximum time for all beacons) or until time t elapses (timeout) and (c) Mousedown, which involves tracking mouse down events then logging the clicks and form submits (Kohavi *et al.*, 2010).

By tracking clicks and submits, Pan *et al.* (2004) concluded that when users view web pages, their behaviors are driven by the gender of subjects, the order of web pages being viewed, and the interaction between site types and the order of the pages being viewed.

2.5 Mouse Activities, Keyboard Activities and Document Scrolls and Their Effects on Gaze Region

2.5.1 Mouse activities and Gaze Tracking

The most common interaction method in today's personal computers is the WIMP (window, icon, menu and pointer). WIMP interfaces are mainly operated using spatial input device usually a mouse. During interaction, mouse cursor travels around the entire interface, switching its functions from pointing, to selection, to drawing, to scrolling, to opening and to jumping, according to what virtual devices (widgets), such as the main document/window, a menu, a scrolling bar, an icon or a hyperlink (Zhai, Smith & Selker, 2007).

Over the years, the mouse has gained great popularity as an input device because of its simple to use mechanisms and accuracy of input. In web based interfaces, it is a major interaction mode that can be used when defining a user model if all mouse clicks and movements on a page are used as an additional layer of information for inferring user interest (Jeff Huang *et al.*., 2012).

Able-bodied users use the mouse for pointing and selection in everyday computing tasks. During web navigation, the mouse is vital for clicking on links. Nearly everyone used the mouse rather than the keyboard to click on links while surfing the Web. Other tasks accomplished using a mouse include launching applications either

from the desktop or the start menu, navigating through folders, minimizing, maximizing and closing applications, moving windows, positioning the cursor when editing text, opening context sensitive menus and hovering over buttons/regions to activate tool-tips among other possible activities. While performing activities, the basic operations performed on mouse to accomplish actions are single click, double click, right click, mouse-over, and click-and drag (Kumar, Paepcke & Winograd, 2007).

Common mouse activities include mouse clicks and mouse movements. In web environment, these activities can only be captured in web search when mouse is inside the browser window. Therefore, when mouse cursor goes beyond the main HTML page and scroll bars (both horizontal and vertical) the mouse is considered as having left the browser window (Schneider *et al.*, 2008).

There are general mouse cursor behavior patterns such as reading, hesitating, scrolling and clicking which can exist as users performs a task on the web, and each pattern has a different meaning. More specific behaviors include users moving the cursor over text as they read, moving the cursor slowly when thinking and tossing aside the cursor to reveal content that the cursor or tooltips were obscuring. These behaviors change the context in which they are interacting with the page (Huang, 2011).

Other identifiable mouse patterns include neglecting the cursor while reading, using the cursor as a reading aid to follow text (either horizontally or vertically), using the cursor to mark interesting results, reading by tracing text, hesitating and reading with the cursor, hesitating before clicking, resting the cursor on white space, ignoring the cursor, examining the page using the cursor, following the text with the cursor and using it to interact with the page or browser (Chen & Lim, 2014).

Interactions with mouse cursor can be recorded with minimal intrusion using JavaScript, which is in-built into all modern Web browsers. Through this, small snippets of JavaScript can be injected on the search page to record cursor interactions and send the data back to the search system using a GET request or HTML5 Web

Sockets. This technique can be deployed on a large scale without disrupting the searcher, or requiring any software installation. Thus, tracking cursor interactions can be instrumented cheaply in commercial search engines or in large-scale user studies on the Web. Cursor movements can thus be recorded using JavaScript at fine levels of detail but the exact resolution depends on the Web browser and operating system. The collated cursor data can then be used to diagnose usability issues at an individual level when the session is replayed. Furthermore, the data can be analyzed in aggregate to generate heat maps. Through JavaScript snippets other cursor activity such as scrolling, highlighting text, or non-navigational clicks may be explored (Huang, 2013).

A good example of using JavaScript to gather mouse data is the Poor Man's Eye tracker Tool for Active Math that tracks students' activities in an e-learning system to obtain data that can be used to diagnose and to react appropriately to the student's actions. In web based systems, JavaScript tracking code can be used to collect data about mouse clicks and movements and keyboard input. Visualizing collected data, mouse trails shown in Figure 2.3 can be identified. Other than mouse data, we can use JavaScript inside the browser, to collect data about keyboard and browser window if input data such as key presses, browser window size is available (Atterer, Wnuk & Schmidt, 2006).

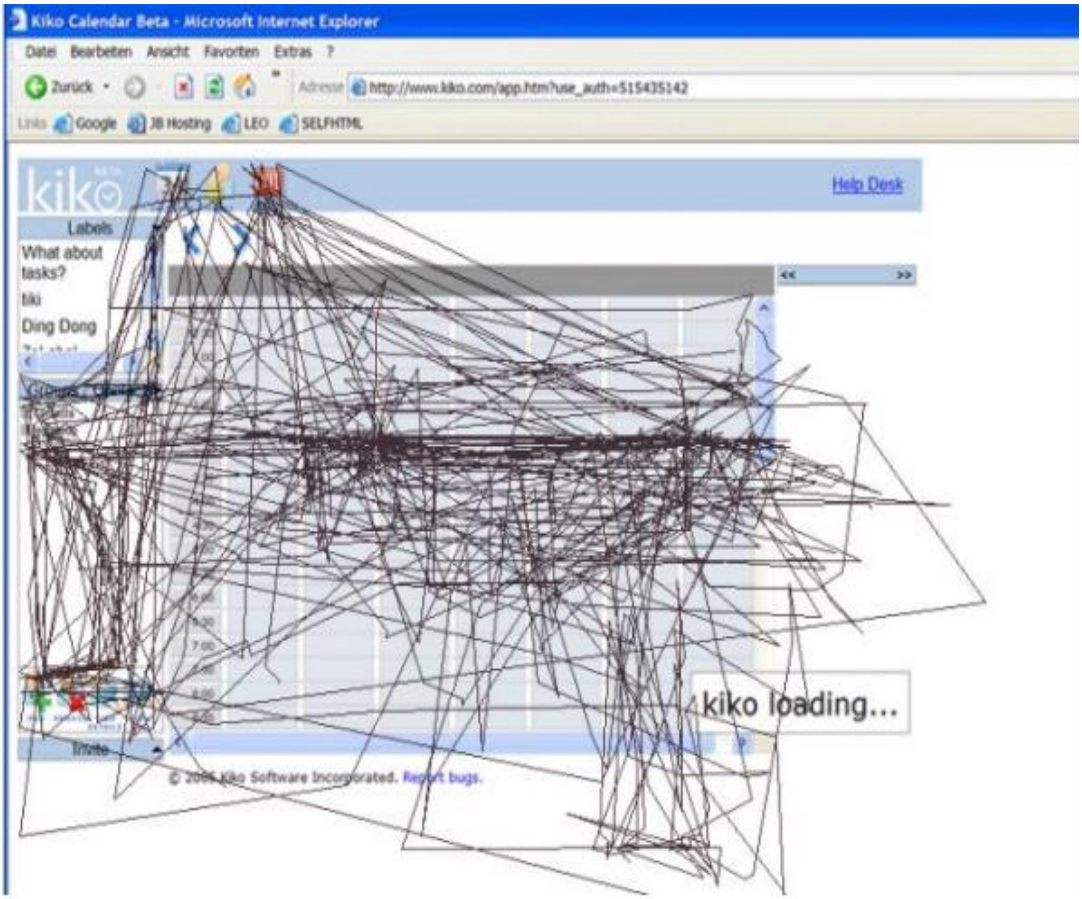

Figure 2.3: Mouse trails recorded by mouse tracker (Huang *et al*., 2011)

In search engine results pages (SERPs) in Figure 2.4, mouse activity data can be identified using heat maps. By examining mouse cursor behavior such as clicks, cursor movements and hovers over different page regions, the cursor position is closely related to eye gaze, especially on SERPs (Huang et al., 2011).

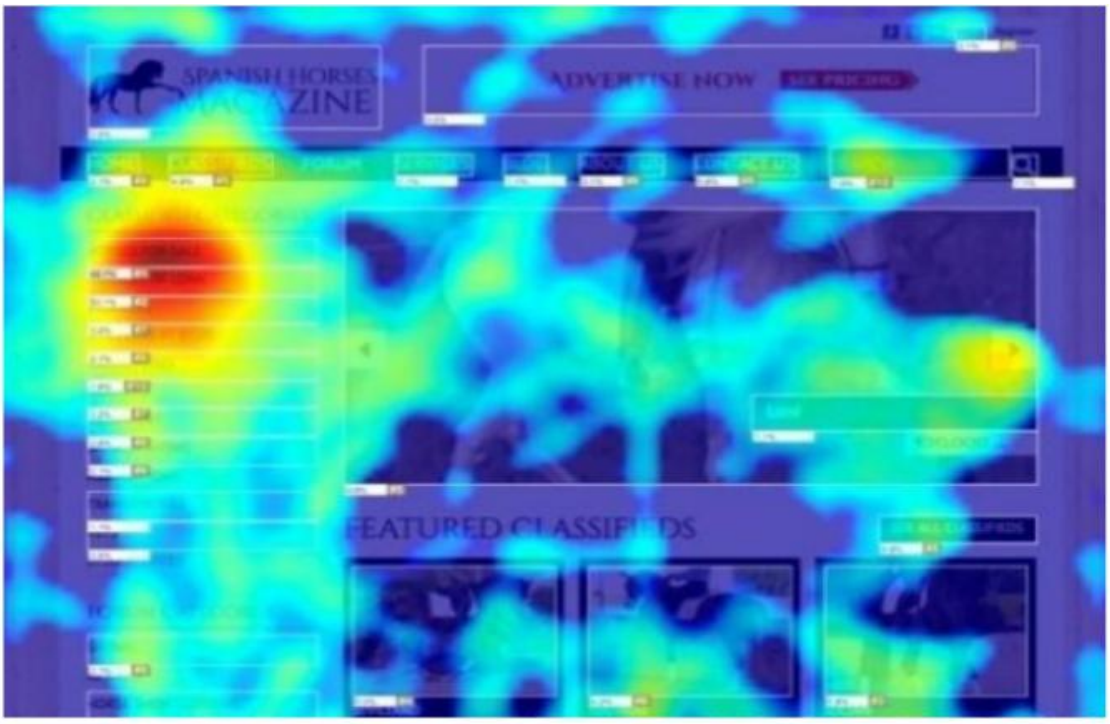

Figure 2.4: The ClickTale generated heat maps (Robu, 2013)

Atterer *et al.* (2006) used a JavaScript tracking code in order to collect data about the users` mouse movements, mouse clicks, hovers, mouse position coordinates, scroll actions and key press actions. Using this method, they found certain areas of the page that were visited by users and even to identify pages desired by a user who had previously left a page.

To visualize the data collected, mouse movement heat maps that show where the visitors looked at the most and clicks heat maps that show exactly where users clicked on the different areas in the website and attention heat maps that show where users focus more on websites can be used (Robu, 2013).

Researchers in the HCI (Human Computer Interaction) field have claimed that the cursor follows the gaze. Chen *et al.* (2001) concludes that there is 84% chance a region visited by the cursor will be visited by the users' gaze. In the work, it was observed that during saccades, the mouse pointer was inside the regions of interest, and possibly used as an instrument which helps the users read and follow the flowing

text. In 70% of the cases, the user's gaze was inside the interest regions except for the blank margins of the page. In their work-Using Cursor Movements to Understand and Improve Search- Huang *et al.* (2012) found out that when using cursor based eye-tracking, cursor and gaze position are correlated. Their research concentrated on mouse clicks and cursor movements where they captured all click positions and cursor movements. They generated the click and cursor movement heat maps shown below.

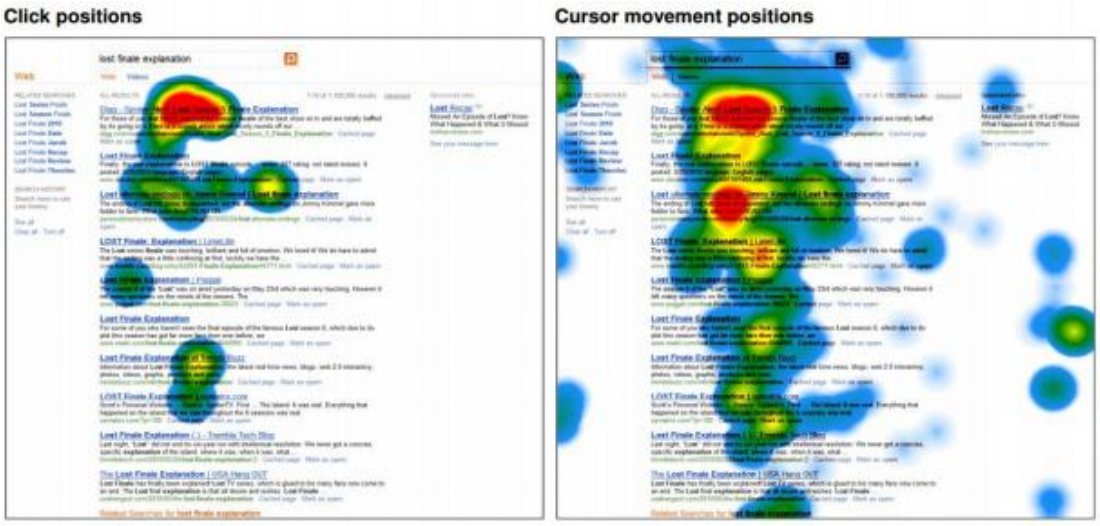

Figure 2.5: Heat maps of click positions recorded cursor positions (Huang *et al.*, 2012)

In the Figure 4, heat maps of all click positions (left) and cursor positions (right) are shown. Heavy interaction occurs in the red/orange/yellow areas, moderate interaction in green areas, while takes place in the light interaction in blue areas. They concluded that although cursor movements, scrolling, and other client-side interactions are easy to collect at large scale, the cursor estimate of the gaze is misguided because it is often not the case depending on time and behavior. It is, therefore, important to predict the real location of the attention when an eye-tracker is unavailable. The correlation of eye-gaze and mouse cursor positions have led to the conclusion that cursor interactions can be useful in determining user intent and relevant parts of Web pages (Huang *et al.*., 2012).

Gaze and cursor in web pages may relate when the distance between them is markedly small in regions where users visit and when the cursor is in motion and in sessions comprising a higher proportion of motion. In search tasks, there is a strong alignment between cursor and gaze positions when the cursor is placed over the search results (Huang, White & Buscher, 2012). However, when a user uses a mouse during search task, the mouse cursor interactions are not all the same because different types of cursor behavior patterns such as reading, hesitating, scrolling and clicking-each of which has a different meaning- exist. This is because various factors may affect search task on the gaze-cursor alignment including time, behavior patterns, and the user. The gaze and cursor positions are better aligned when current gaze position is compared to a future cursor position (Huang *et al.*, 2012).

There are general mouse cursor behavior patterns such as reading, hesitating, scrolling and clicking which can exist as users performs a task on the web, and each pattern has a different meaning. More specific behaviors include users moving the cursor over text as they read, moving the cursor slowly when thinking and tossing aside the cursor to reveal content that the cursor or tooltips were obscuring. These behaviors change the context in which they are interacting with the page (Huang, 2011).

2.5.2 Keyboard activities and gaze region

Common keyboards such as QWERTY keyboard, split-angle keyboards, Microsoft Office Keyboard and the Xerox Star keyboard enable users perform tasks in a computer that requires data input from the user (Claypool *et al.*, 2001). The following features are included in common computer keyboards:

- A keyboard can be used as a scrolling device. It gives one the ability to scroll within a document as well as moving between open applications and windows within an application. This is achieved using dedicated keys as well as key combinations such as ALT + TAB.

- The keyboard can also be used to Cut, Copy and Paste data within applications, between applications and copying, moving and deleting application and files within the file system.

- One can also perform common actions such as navigation within a document by moving back or forward, performing Undo, Redo, Print, Save, Help, Send, Reply, New, Open, and Close actions.

- Experienced users may use the keyboard for launching applications using dedicated launch keys and key combinations. These users may as well use editing keys (Home, End, Delete, etc.) to rearrange a document while editing.

Many operations performed by mouse can also be performed using the keyboard. Therefore, most people use the keyboard when they are not using the mouse but rarely both at the same time (Claypool *et al.*, 2001). Keyboard strokes monitoring is therefore critical in tracking user activities and the following activities can be discovered:

Text typing- Computer users enter a lot of information through the keyboard such as creating email messages, instant messages, documents, and code among others. Entering or typing text is achieved through pressing keyboard keys which define the characters the user is entering (Balzarotti, Cova & Vigna, 2008).

Menu activation - Menus can be activated using keyboard through simple keystrokes such as Alt + F thus enabling navigation through the menu by using arrow keys.

Text selection - The keyboard can be used to perform text selected commonly done using a mouse. This is usually done by focusing [cursor] on text to be selected then using Shift + right arrow and Shift + left arrow keys.

Activation of desired actions - Users can use the keyboard to activate desired actions if hot keys are created to do so. For example, a majority of traditional computer keyboards have the following activated function keys: F1 to F12 for short-cuts or "hot keys", Alt+Tab to "switch applications" and Ctrl+C for "copy".

To terminate an action the user can look away or anywhere outside of the zoomed region and release the hotkey, or press the Esc key on the keyboard. Special keys like the Esc key on the keyboard can be used terminate an action the user. The hotkey-based triggering mechanism makes it simple for able-bodied users to keep their hands on the keyboard to perform most pointing and selection operations (Kumar, Paepcke & Winograd, 2007).

A number of studies have shown some relationship between the keyboard and gaze region. Firstly, in gaming, the keyboard together with mouse and an eye tracker give the orders for target action through the keyboard. The gamer selects a target with the mouse. In 2-D scenes, keyboard keys can be used for navigation where users scroll with the cursor keys on the keyboard while in most modern computing devices, on-screen keyboards are provided for text entry is necessary (Isokoski, Joos, Spakov & Martin, 2009). Secondly, in keyboard-controlled games, a player controls the game character by pressing cursor keys while scanning the visual field (Krejtz *et al.*, 2014).

Lastly, the EyePoint technique illustrated in Figure 2.6 for pointing and selection uses a combination of eye gaze and keyboard triggers for progressive refinement of target using look-press-look-release actions. In the process, the user first looks at the desired target then by pressing and holding down a hotkey, the user brings up a magnified view of the region the user was looking at. The user then looks again at the target in the magnified view and releases the hotkey to perform the mouse action (Kumar, Paepcke & Winograd, 2007).

Figure 2.6: EyePoint technique for pointing and selection

2.5.3 User document Scrolls activities and gaze region

When users read through lengthy web documents, scrolling becomes a necessity because long documents cannot be displayed within a single view conveniently (Zhai, Smith & Selker, 2007). Scrolling involves acquisition of targets well beyond the edges of the screen and users scroll manually through long documents to find and view content of interest in the window. As a result, scrolling provides a simple means for navigating through long documents that are too large to be displayed within a single view in a convenient manner. Scrolling can be achieved through input devices such as scroll wheels, which are provided on most pointing devices.

Hinckley *et al.* (2015) noted that scrolling is influenced by two main factors-scrolling distance and scrolling tolerance. In another study by Goecks and Shavlik (2015), the amount of scrolling performed by the user is determined by the number of command-state changes on the page. A command-state change is an internal event in the IE event model that occurs when the user re-sizes the IE window, utilizes the Edit menu, or scrolls. The vast majority of the command-state changes are the result of user scrolling, and thus command-state changes serve as a reliable surrogate for the amount of user-scrolling activity.

The main tool used during scrolling is the mouse cursor and a scroll bar. If a user is working on a document and the point of interest moves to outside of the viewing window, he/she is forced to scroll the document to sections hidden from main view

window. Scrolling may also occur automatically when text cursor is available and users navigate to sections of documents below the bottom viewing window. Scrolling a document can be achieved using three main methods: i) Use of scroll bar, where scrolling is achieved by acquiring the scroll bar handle and drag it; ii) Use of the mouse cursor to press the arrow buttons at the ends of the scrolling bar, causing the document to scroll; and iii) clicking on the rest of the space on the scroll bar, causing the document to "jump scroll" (Zhai, Smith & Selker, 2007).

In addition to the scrollbar, scrolling can also be achieved using a mouse scroll wheels (Huang, White and Dumais, 2011). Most scrolling devices are linear in nature, which makes scrolling achievable only over a linear distance either vertically or horizontally. There are two main activities that can be performed on documents when scrolling with a pointing device- short and long range movement scroll. In short range movement scroll devices are best used only for short range movement whereas long-range movement scroll is experienced when users scroll long documents (Huang, White and Dumas, 2011).

When longer movements are required, users can switch to an alternative control such as dragging the scroll thumb as part of a scrollbar but this is slower compared to shorter movements. However, despite most scrolling devices allowing faster scrolling in short documents, they become inefficient for long distance scrolling. This limitation notwithstanding, the performance of devices such as scrolling wheel can be improved significantly using an acceleration algorithm (Cockburn *et al.*, 2012).

User scrolling interactions events data has been found to be predictive and may indicate user interests when the target is visible and the user scrolls a short distance. Such scrolls have also been classified as analogous to target selection using an area cursor (Hinckley, 2015). Scrolling, cursor movement and hovers have in his respect been used to accurately infer search intent and interest in search results (Guo & Agichtein, 2010). Buscher *et al.* (2009) used scrolling to infer user interest and compared it to gaze tracking feedback through the Poor Man's Eye Tracker. This tracker employed cursor tracking to approximate gaze tracking without the eye

tracker. Depending on the accuracy required, such a tracker can be used to predict user interest (Cooke, 2006).

In SERP, most of the users' attention is still focused on the top ranked results which are visible before scrolling down (Joachims *et al.*, 2005). However, when viewing a SERP multiple times, a user's attention tends to shift to focus more on results at the bottom of a web page (Jiang & Allan, 2015). Claypool, Mark *et al.*, (2000 identified that the amount of time spent on a page together with amount of scrolling are correlated with explicit interest on web page.

Liebling and Dumais, (2014) established that when the mouse acquires the scroll bar, successive clicks on the bar occur as the eyes remain in the reading area without any additional visual control. Goecks and Shavlik (2015) discovered that the amount of user scrolling together with the number of hyperlinks clicked and the amount of mouse activity can be used to gauge the importance of a web page to the user. Thus, scrolling may affect user eye gaze region on the screen.

Scrolling behavior and information on the browser's viewport can be used effectively as gaze tracking feedback in search scenarios. Scrolling may cause the cursor to move down relative to the page, making it a good estimate of vertical attention (Huang *et al.*, 2012). To visualize scrolling in a web browser, scroll-heat maps have been used to show how far visitors scroll. The results help web page developers to decide if pages should be shorter or not (Robu, 2013).

Apart from scrolls, various user actions appear likely to correlate with the user's interest in a page. These include the number of hyperlinks clicked on by the user, and whether the user bookmarked the page or not (Goecks & Shavlik, 2015).

2.5.4 Eye/Gaze region Estimation Methods and Techniques

To estimate eye gaze, both regression and machine based learning methods have been used. Zhu and Yang (2002) used a 2-D linear mapping from the vector between the eye corner and the iris center to the gaze angle to detect the inner eye corner and the center of an iris at sub-pixel accuracy. They estimated gaze directions through

interpolation using the gaze angle and the vector from the eye corner to the iris center. This was achieved by constructing a 2-D linear mapping created from the vector between the eye corner and the iris center to the gaze angle. The gaze directions in consecutive frames then calculated through interpolation. For example, suppose the gaze angle and the vector from the eye corner to the iris center for calibration point P_1, P_2 are respectively $\{(\alpha_1, \beta_1), (x_1, y_1)\}$ and $\{(\alpha_2, \beta_2), (x_2, y_2)\}$, then if we observe a corner-iris vector *(x, y),* the corresponding gaze angle is calculated using equation 1 and 2:

$$\alpha = \alpha_1 + \frac{x - x_1}{x_2 - x_1}(\alpha_2 - \alpha_1) \quad (1)$$

$$\beta = \beta_1 + \frac{y - y_1}{y_2 - y_1}(\beta_2 - \beta_1) \quad (2)$$

This approach is advantageous in that:

i. It is conceptually simple and efficient.

ii. It is extremely accurate as the only approximation employed is a simplification of the rotation angle of the eyeball as the projection of the movement of the iris center (Zhu & Yang, 2002).

Huang, white and Buscher, (2012) estimated gaze in search logs by recording gaze positions approximately every 20 ms using Tobii x50 eye tracker and recording cursor positions using a client side script approximately every 100 ms then interpolating gaze position for every cursor position. Interpolation involved computing gaze x and y coordinates weighed by the coordinates of nearest gaze coordinates. The computation was done as follows.

Since cursor and gaze events did not necessarily have identical time stamps, a gaze position was interpolated for every cursor position. Interpolation was performed by computing gaze x and y coordinates weighted by the coordinates of the nearest gaze coordinates before and after the cursor position (Huang, White & Buscher, 2012). The interpolated x-coordinate for eye gaze is computed using equation 3.

$$x_i = x_0 + (x_1 - x_0)\frac{t_i - t_0}{t_1 - t_0} \quad (3)$$

$$y_i = y_0 + (y_1 - y_0)\frac{t_i - t_0}{t_1 - t_0} \quad (4)$$

Where t_i is the time for the corresponding cursor position, x_0 is the gaze's x-coordinate preceding the cursor position, recorded at time t_0, and x_1 is the gaze's x-coordinate following the cursor position, recorded at time t_1. The interpolated y-coordinate was computed the same way, by substituting x for y in the above equation 4. To reduce interpolation inaccuracies due to noise from the eye-tracker, cursor positions were only captured if they occurred between gaze positions that were at most 100 ms apart (Huang, White & Buscher, 2012) .

In his work, Modeling User Behavior and Attention in Search, Hung, J (2013), used a linear regression model of interaction features to estimate the gaze position of the subject under investigation to determine whether the estimated position was closer to the ground truth than the simple cursor position. The ground truth is the gaze position measured by the eye-tracking system during the lab study. The x- and y-coordinates

of the gaze position were estimated separately (i.e., univariate prediction). To compute the weights (coefficients) for each feature, a multiple linear regression was performed as demonstrated by the model below. The model illustrates the value of gaze prediction in an example query session with cursor positions, gaze positions, and predicted gaze positions overlaid on the SERP (Huang, 2013). The model for the regression for the x-coordinate used in this case is shown in equation 5 below:

$$g_x \sim c_x + \log(t_d) + \log(t_m) + c_x \times \log(t_d) + c_x \times \log(t_m) + f_x \qquad (5)$$

Where:

g_x : x-coordinate of gaze position

c_x : x-coordinate of cursor position

t_d : dwell time

t_m : time since movement

f_x : most likely coordinate of the gaze based on future cursor position.

Steichen *et al.* (2013) demonstrated that a user's eye gaze is a valuable source to infer a number of task and user characteristics from using simple machine learning techniques on simple eye tracking metrics. They identified that using information on user eye gaze patterns while interacting with a given visualization, one can predict the user's visualization tasks as well as user cognitive abilities including perceptual speed, visual working memory, and verbal working memory using adaptive visualization systems. They also showed that predictions made are significantly better than a baseline classifier. They focused on designing visualization systems that can adapt to each individual user in real-time.

Iqbal and Bailey (2004) identified that eye gaze patterns can be used to classify the type of task the user is performing in real time and that this parameter can inform

decisions about when to interrupt a user engaged in a primary task. However, understanding user needs is challenging when it involves assessing the user's high-level mental states as these are not easily reflected by interface hence the need for Intelligent User Interfaces (IUI).

IUIs can be used to learn a user's needs and act accordingly in order to improve the user's interaction experience. Within Intelligent Learning Environment (ILE) systems an IUI can provide personalized instruction, just like human educators do to accommodate individual students' needs by personalizing the instruction to target the meta-cognitive processes relevant to learning. In Adaptive interfaces on the other hand, eye-tracking data on user attention are useful when modeling the user's high-level mental processes (Conati *et al.*, 2007).

Eye tracking devices have been used to collect user data on eye movements with a view to understanding where users look or focus their attention during HCI. Nonetheless, alternatives to eye tracking devices exist- key among them being visual salience models capable of predicting fixation locations. These models are evaluated against ground-truth eye-tracking data collected from human subjects. When using the models machine learning that train image features from eye trackers can be used to improve gaze/fixation estimation. In this regard, improvements in salience detection have taken place through the use of features such as a simple dynamic salience model and a decision tree classifier (Judd *et al.*, 2009).

In Interactive Learning Environments (ILE) such as the Adaptive Coach for Exploration (ACE), eye-tracking data from gaze patterns helps assess student reasoning on individual exploration cases and consequent effectiveness of student exploration. Student gaze in ACE can be tracked by head-mounted eye-trackers. With respect to ACE, eye tracking information improves assessment of the effectiveness of student exploration by identifying groups of students with distinct learning proficiency through clustering, as opposed to using interface actions and latency information alone. As a result, eye tracking data reveals student reflection and leads to identification of more complex learning patterns (Amershi & Conati, 2007).

Recently, machine-learning techniques have been applied to gaze data to predict user intents, cognitive processes, student learning, user characteristics and visualization tasks. For example, ANNs have been used to minimize the mouse jitter in recorded eye-gaze behavior by finding the relationship between the gaze coordinates and the mouse cursor position based on the multilayer perception modeling. When using ANNs, an eye-gaze-based HCI system can accommodate and adapts to different users thus guaranteeing a certain level of accuracy in terms of collated data (Steichen *et al.*, 2013).

Regression Neural Networks (RNNs) have also been used in gaze estimation by mapping feature vectors extracted from images to output gaze location of the user. This is achieved through the use of raw eye image pixels as the input into a 3-layer feed-forward ANN for gaze estimation. Based on this, examination of the effect of different visual features on the accuracy of gaze estimation concluded that if extraction of low-level features from images of the eye is done and machine learning is applied to the features, gaze estimation errors are reduced significantly (Yanxia *et al.*, 2012).

Other research works have attempted to automatically track the visual attention in dynamic visual scenes using the Bayesian Learning Algorithm, which identifies visual saccades (transitions) from visual fixation clusters (regions of interest). This approach has been evaluated using real-world data collected from eye-tracking experiments during driving sessions (Enkelejda *et al.*, 2012). Additionally, Support Vector Machine (SVM) learning method has been used for classification of human behavior, especially for detecting cognitive states via eye movement data using machine learning and pattern recognition methods (Eivazi & Bednarik, 2011).

2.7 The Conceptual Framework

This section describes a predictive conceptual model describing the relationship between mouse cursor position, document scrolls and eye gaze region over a period of time. Mouse clicks and drags, document scrolls, key presses and cursor movements, which are the main actions considered when estimating a user's gaze

may influence the gaze region over time depending on user actions when interacting with web pages and other documents. The gaze region of a person can then be computed by identifying gaze positions when an action is performed using these input devices.

Data collected in this regard includes mouse cursor positions upon clicking or dragging the mouse, document position after scrolling, document loaded and loading point on keyboard navigation and the amount of scrolls performed. The User interactions with a web interface as discussed below can be visualized using the following diagram, which demonstrates the relationship between web navigation activities of mouse, keyboard and document scroll and the user's gaze position. A linear regression model was used to estimate gaze region.

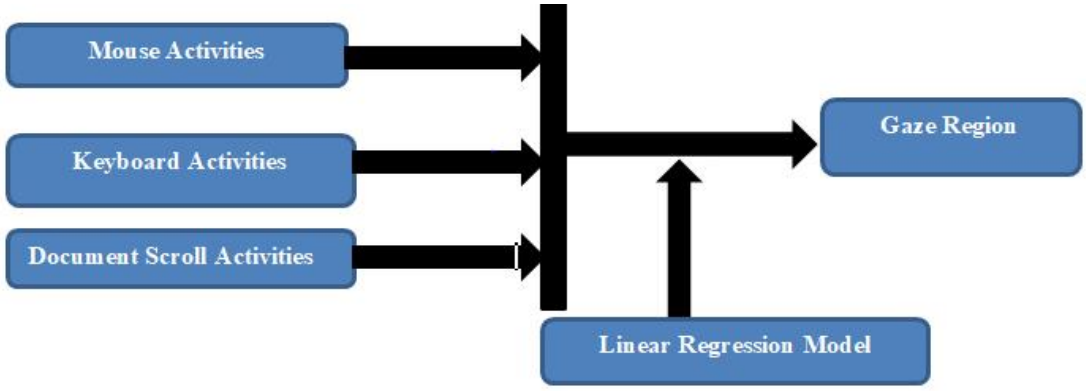

Figure 2.7: Conceptual Framework

2.7.1 Determining gaze position from mouse activities

In regards to mouse data, users can perform two main actions using a mouse-clicking and moving. Mouse clicks occur when a person holds down a mouse button while performing an action. Clicks can be classified as left (default), right or middle clicks (Huang, 2011). A click can therefore be used to indicate a gaze fixation since users click links or items of interest as shown in the diagram below.

Figure 2.8: Mouse click at position X1, Y1 considers X1, Y1 as gaze position at time t_i

Users also move mouse to areas of interest within a page either as a reading guide or to perform clicks on links or other items on a document. Thus, when mouse moves from one point to another, the last position after movement can be considered as the current gaze fixation position.

Figure 2.9: Mouse cursor moving from X_1, Y_1 at time t1 to X_2, Y_2 at time t_2 will consider X_2, Y_2 at current gaze fixation at time t_2

2.7.2 Determining Gaze position from document scroll activities

At times, a user may scroll document while the cursor is stationary especially when keyboard arrow navigational keys are used. In such cases the document viewpoint changes as the cursor remains stationary. Sometimes the cursor may end up invisible from the view point and this renders the cursor irrelevant as an indicator of gaze fixation position. To determine possible gaze fixation position after a document scroll, we can calculate possible vertical mouse position using gathered scroll information. The amount of document scroll and last mouse cursor position can as a result be used to determine possible gaze locations. Figure 2.10 illustrates scrolling vertically from position Y_1 to Y_2 using scroll amount from Y_1 to Y_2 to compute possible mouse position at Y_2 in reference to mouse position at document position Y_1. From literature, we learnt that scrolling can be achieved using three main methods: i) Use of scroll bar, where scrolling is achieved by acquiring the scroll bar handle and

drag it; ii) Use of the mouse cursor to press the arrow buttons at the ends of the scrolling bar, causing the document to scroll; and iii) clicking on the rest of the space on the scroll bar, causing the document to "jump scroll" (Zhai, Smith & Selker, 2007).

Figure 2.10: Document scrolling vertically

Most scrolling devices are linear in nature, which makes scrolling achievable only over a linear distance either vertically or horizontally (Huang, White and Dumais, 2011). Scroll amount helps us compute the scrolling distance Hinckley *et al.* (2015).

When a cursor insertion point is available and a user scrolls documents using keyboard arrow keys, the mouse position is considered to be the current cursor insertion point. That is, moving from X_1, Y_1 to X_2, Y_2 will consider X_2, Y_2 as the current gaze position (Isokoski, Joos, Spakov & Martin, 2009). This is illustrated below.

Figure 2.11: Cursor Insertion Point

2.8.3 Determining Gaze position from keyboard activities

If one inputs text using the keyboard, gaze is expected to switch to the text input control where text is input. Kumar, Paepcke and Winograd (2007) elaborate that

while using keyboard, user first looks at the desired target before pressing the keys. The user then looks again at the target and releases the pressed key. This will result to gaze position remaining at X_1, Y_1 which is the position of the element on the page document. Figure 2.12 illustrates this.

Figure 2.12: Using keyboard to input text

If a user makes use of keyboard navigation keys to load and view dynamically loading content, the center location X_1, Y_1 will be considered as gaze region.

Figure 2.13: Using keyboard navigation keys

CHAPTER THREE

METHODOLOGY

3.1 Introduction

This chapter illustrates how we can estimate gaze region more accurately on web pages using additional source of user input data. Rather than estimate gaze using only cursor interactions, we will consider all useful activities generated by a mouse. Additionally, we will consider activities generated by the keyboard. This leads to a useful interaction activity, including scroll, which can be generated by either the mouse or the keyboard.

In his work on eye-gaze position prediction, Huang, 2013 found out that one can actually predict the eye-gaze position based solely on cursor interactions, so that visual attention can be determined at any point in time, and at scale to better understanding of searcher behavior

To estimate gaze position, one must carefully understand the layout of web pages. Web pages are made rectangular window composed of an inverted Cartesian with (X, Y) coordinated. The (0, 0) coordinated is at the top-left part of the page. Top-left to the right we have the x values and top-left to the bottom we have the y values. To get a position on page we read right for the value of x and down for the value y. (X, Y) values are measured in pixels in most cases. REMOVE

Web pages consist of one or more elements which include paragraphs, divisions, lines, drawings and other user input elements such as textboxes, checkboxes, and radio buttons and so on. Each element has an (X, Y) position placement. Users interact with these elements via input devices such as keyboard, mouse and track pads. The interaction require user's active participation over a period of time when they are in a session. During interactions, users manipulate the input devices using provided controls to active some tasks on the web interface. The behaviors of these input devices therefore have a strong influence on gaze region because users may not

manipulate them “blindly”. Huang showed that there is effect of cursor on gaze position.

The section begins by outlining the research design followed by a discussion of method employed in data collection. Finally, data analysis methods used and expected output will be discussed.

3.2 Research Design

During the study, empirical research methods are adopted with a view to collecting empirical observations and/or data in order to answer particular research questions.

The study involved development web pages consisting of both open-ended and closed questions that allowed respondents to have a typical interaction with a web forms while answering. Questions were developed using Lime Survey, open source on-line statistical survey web app and a JavaScript code embedded on it so as to monitor participants’ activities. Sample questions are in appendix VI. Each participant had a unique session and the data was aggregated as per the sessions.

Users interacted with web page elements via input devices such as keyboard, mouse and track pads. The interaction required user’s active participation over a period of time when they are in a session. During interactions, users manipulated the input devices using provided controls to active some tasks on the web interface. Mouse activities, keyboard activities and document scroll information were monitored while users interacted with web page elements.

Thus, the main tool used in data collection was a web application. The application provided both static and dynamic content and was hosted in a web server then accessed via a web browser. The application consisted of two main modules, the views consisting of web pages and an application programming interface (API) server to serve requests by the users. Client views consisted of HTML web pages and a special script designed to listen to mouse, keyboard and scrolls by the respondents.

When a user loaded a web page in a browser, a script was used to collect data that was then pushed to the server for recording in a database for later analysis.

The server side consisted of Representational state transfer (REST) application programming interface (API) which is a web service. The API was a server side PHP script that communicated with the browser script. The server also hosted a relational SQL database which was used to record data received by the service from browser clients. The diagram in Figure 3.1 below depicts the system architecture.

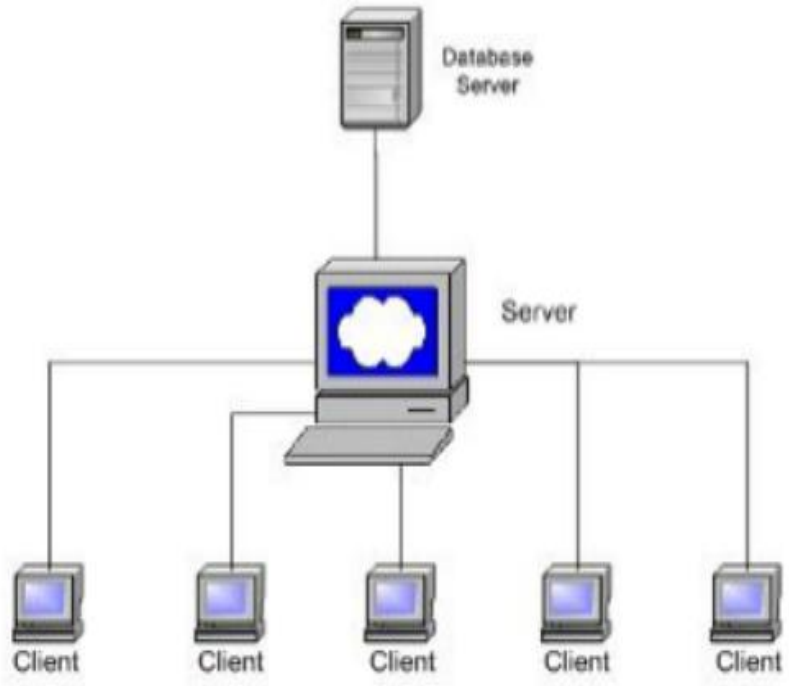

Figure 3.1: Client Server Architecture (Rose India, 2014)

3.3 Population

The study targeted B.Sc. IT 3rd year students in JKUAT. The total population of students were one hundred and twenty (120). This was the total number of students enrolled during the January - April 2017 semester.

3.4 Population Sampling

During the study census sampling method was adopted so that all respondents in the population were used.

3.5 Data Collection Procedure

This work captured and analyzed mouse activities, keyboard activities and document scrolling activities. Every respondent who visited the page had to perform assigned tasks that required them to use mouse or keyboard or both. While the respondents performed tasks a script monitored their mouse and document scrolls activities. For accuracy and consistency, the selected population sample were registered for identification. Once registered individuals logged in to the website, their activities were tracked.

The capturing process involved browser script that captured the x and y coordinated of mouse as it moved on the screen in reference to top-left corner of browser window (coordinate (0,0)). Collected data was classified based on events users performed. Mouse events captured were : mouse move, mouseup, click, dblclick, contextmenu, wheel, cut, copy, paste, dragstart, dragenter, drag, dragover, dragleave, dragend, select, mouse wheel and mouse scroll. Mouseleave and mouse enter were treated as mouse move. We were not interested in mouse key release because they may not have signified click or double click. Instead, mouseup was considered. For each mouse event, the mouse position at which event happened was also captured. Keyboard events captured included: keydown, keypress and keyup. Browser script codes for data collection are illustrated in Appendix V.

For each keyboard event, the mouse position at which event happened was also captured because the mouse position is always available throughout the session. Scroll events emanated from keyboard or mouse activities. They were processed when scroll keys were pressed and when mouse wheel and mouse dragging events were detected. Recorded information about scroll included, scroll amount and cursor insertion point when available. For each scroll event, the mouse position at which

event happened was also captured because the mouse position is always available throughout the session. Time when each activity took place was also recorded.

Data collection was done at fixed time intervals of 300ms then pushed to remote server for storage after every 2000ms for later analysis. Session for each user was created and scored in the browser as a cookie. The session was then destroyed when users reached end of the questions.

3.6 Data Analysis

To estimate gaze position, we must carefully understand the layout of web pages. Web pages are made rectangular window composed of an inverted Cartesian with (X, Y) coordinated. The (0, 0) coordinated is at the top-left part of the page. Top-left to the right we have the x values and top-left to the bottom we have the y values. To get a position on page we read right for the value of x and down for the value y. (X, Y) values are measured in pixels in most cases. Web pages consist of one or more elements which include paragraphs, divisions, lines, drawings and other user input elements such as textboxes, checkboxes, and radio buttons and so on. Each element has an (X, Y) position placement.

We use a linear regression model work to estimate the average gaze position. The true gaze position will be coordinates recorded by user activity tracking system during lab study. The estimated (X, Y) gaze position will be done separately using a linear regression model.

Before analysis, data was collected via a browser based script described in the previous section to capture relevant features for use in the experiment. Collected data was stored in the following format.

Table 3.1: Data collected via browser script

Table Column	Data Type	Description
sessionkey	bigint(20)	Unique key for each user session
element	varchar(200)	Type of element user is interacting with e.g. paragraph, divisions, push buttons, input field, radio button, check box and so on
event	varchar(200)	The type of event generated either by mouse, keyboard or scroll
mousePosition	varchar(50)	Current mouse position which is available throughout the session
elementPosition	varchar(50)	The position of current element which the user is interacting with if any
elementMousePosition	varchar(50)	The position of mouse within the current element which the user is interacting with.
docscrolltop	varchar(50)	Position of scroll when user stops scrolling the bar if any
docscrolldelta	varchar(50)	The amount of scroll user does measured in delta
elemscrolltop	int(11)	Position of scroll inside an element when user stops scrolling the bar if any
elemscrolldelta	int(11)	The amount of scroll user does inside an element measured in delta if any

During data analysis, data about user activities collected and time of collection were analyzed to determine possible gaze position per page viewed. Analysis was done in two main phases:

Phase 1: Classifying data into mouse activities, keyboard activities and scroll activities.

Phase 2: Estimating Gaze using classified

3.6.1 Classifying data into mouse activities, keyboard activities and scroll activities

During this phase, collected data and stored in the previous section was classified to keyboard activity, mouse activity or scroll activity depending of the input event captured during collection. Classification involved analysis of mouse, keyboard and scroll data. For mouse data, we collect the mouse X and Y position as the gaze positions for mouse activities. For keyboard activities, we collect the element X and Y position as the gaze positions. Each element has X and Y position on the web page. When scroll activities are encountered, we collect the scroll top position which gives us the Y position as the gaze positions. Since scroll is mainly vertical, we assume the X position does not change. Thus the last X gaze position is used. See flow chart in Figure 3.2. The java codes for the algorithm are available in appendix III. This process can also be achieved using the following pseudo-code:

```
Start

Input an event

If an event is a mouse_scroll or an event is a keyboard_scroll or an
event is an elem_scroll_stop or an event is a document_scroll then
an event is a scroll_activity

Else if an event source mouse then an event is mouse_activity

Else if an event source keyboard then an event is keyboard_activity

End
```

The first test was important because some event such as context menu, cut and paste can emanate from either keyboard or mouse. Thus such events form the mouse were isolated first. The algorithm starts by selecting unique user data using special session key.

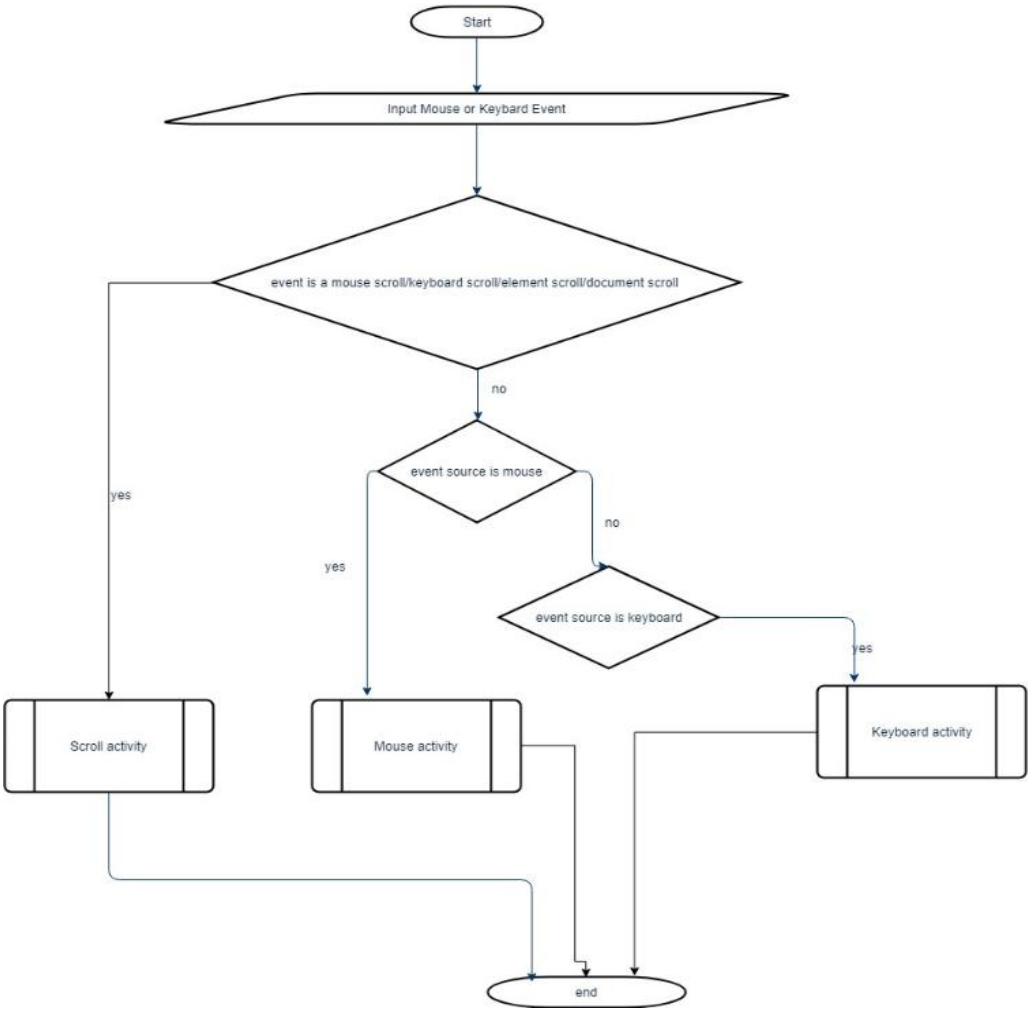

Figure 3.2: Classifying data into mouse activities, keyboard activities and scroll activities

3.6.2 Estimating Gaze with Mouse, Keyboard and Scroll Interactions

The subject's user's gaze can be estimated using Lindeman, Merenda and Gold (LMG) metric linear regression model to determine whether the estimated position is nearer to the ground truth, cursor position. The metric determined the relative importance of the features for predicting the x- coordinate and y- coordinate. This metric was used by Huang et al., 2011 to predict cursor position using collected cursor data. The LMG metric linear regression model in equation 6 and equation 7 below was use to analyze keyboard, scroll and mouse activities:

$$g_x \sim c_x + \log(t_d) + \log(t_m) + c_x \times \log(t_d) + c_x \times \log(t_m) + f_x \quad (6)$$

Where :

t_m is the time since a last gaze position

t_d is the dwell time

c_x is the x-coordinate of the gaze position

g_x is the x-coordinate of the gaze position

f_x is the most likely x coordinate of the gaze position based on future gaze positions.

Using the model, we can predict gaze x position from collected data. The regression equation for the y-coordinate was similar, but substituting *x* for *y* in regression model as shown in equation 7.

$$g_y \sim c_y + \log(t_d) + \log(t_m) + c_y \times \log(t_d) + c_y \times \log(t_m) + f_y \quad \text{.....(7)}$$

Where :

t_m is the time since gaze position

t_d is the dwell time

c_y is the y-coordinate of the gaze position

g_y is the y-coordinate of the gaze position

f_y is the most likely y coordinate of the gaze based on future gaze positions.

In the model, input data was from mouse, keyboard and document scrolls activities and the time of information capturing. Using algorithm pseudo code in figure 3.3, most probable gaze position was estimated from the input sources discussed. The java codes for the algorithm are available in appendix IV.

```
Step 1: Start

Step 2: Input user session key and create variables event of type
string, gaze of type list<integer, integer>, mouse position gaze
of type list<integer, integer>, gaze X position of type integer,
gaze Y position of type integer

Step 3: Select activity data for session key from database
ordered by tracking time Ascending

Step 4: Initialize gaze(X, Y) List to new Instance of Array List

Step 5: Repeat the steps below until activity data is null

Step 6: Initialize mouse position to current activity data mouse
position
```

Step 7: Initialize event to current activity data event

Step 8: Set Gaze X position to mouse X position

Step 9: Set Gaze Y position to mouse Y position

Step 10: Check the event captured to see if it is a keyboard event

Step 10.1: Check the activity for event captured to see if it is a keyboard scroll. When the activity is a keyboard scroll, go to step 10.2 otherwise go to step 11

Step 10.2: Check whether document scroll top is greater than zero (Meaning that the user is scrolling using keyboard) then Set Gaze Y position to document scroll top (Gaze X position does not change. Mouse X position is picked)

Step 11: If the activity data is considered as from the keyboard go to **step 11.1** Otherwise go to step 14.

Step 11.1: Check whether the HTML element left position is greater than zero then (Means that the user is probably typing in some text)

Step 11.2: When element left position is greater than zero, set Gaze X position to HTML element left position plus the HTML element Mouse X position (Which is equal to Mouse X position)

Step 12: Check the event captured to find out if it is a Mouse event

Step 12.1: Check the activity for event captured to see if it is a mouse scroll. When the activity is a mouse scroll go to step 12.2 otherwise go to step 13

Step 12.2: Check whether document scroll top is greater than zero (Meaning that the user is scrolling using mouse) then Set Gaze Y position to document scroll top (Gaze X position does not change. Mouse X position is picked)

Step 13: Otherwise the activity data is from the mouse. We ignore because we have already captured Gaze position from mouse Y and mouse Y in Step 6 above.

Step 14: Check whether document scroll stop is greater than zero *(User stopped Scrolling on document).* When the document scroll event is available, Set Gaze Y position to document scroll top and X

```
position is picked from mouse.
Step 15: End
```

Figure 3.3: Algorithm for classifying data into mouse, keyboard and scroll activities

The above algorithm implements the conceptual model discussed at the end of the previous chapter. The algorithm inputs a unique session key that identifies user data. Data for the user session is then loaded from the database for gaze region estimation.

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1 Introduction

This research exercise was undertaken to determine if user activities other than mouse cursor affect gaze region while user manipulates a web page. In previous studies, a user's gaze region was established to be aligned to the mouse cursor. In this study, a web platform, Lime Survey was used to provide intellectual question to a group of students. The students were required to answer the questions by typing in answers, checking radio buttons and check boxes as well as selecting options from drop down boxes.

A JavaScript code was embedded in Lime Survey software to collect usage data as students attempted the questions provided. Data collected included mouse activities such as clicks and moves, and keyboard activities mainly key presses. Scroll activities were extracted from activities of the mouse and keyboard. Collected data was then pushed to a remote server via RESTful URL for storage in a SQL database. Data collected was then analysed and grouped into mouse activities, keyboard activities and scroll activities as discussed in previous section.

Data was further analysed to deduce useful patterns related to mouse, keyboard, scroll and a combination of the three using LMG metric linear regression model. An algorithm that takes into consideration the captured user activities in conjunction with LMG metric was developed to estimate a probable gaze region in the form of (x,y) coordinates from the data captured.

In this study, mouse clicks and drags, document scrolls, key presses and cursor movements which are the main actions may influence the gaze region over time depending on user actions when interacting with web pages and other documents. The gaze region of a person can then be computed by identifying gaze positions when an action is performed using these input devices.

Using the algorithm described in chapter three, we can easily deduce the following:

i. Percentage comparison of time spent using mouse, keyboard and scroll per session

ii. Path users followed when using mouse, keyboard and scroll per session

iii. Estimated path followed when using mouse keyboard and scroll

iv. Comparison of estimated path in (iii) with ground truth mouse path during the study.

4.2 Descriptive Analysis

4.2.1 Percentage comparison of activities and time spent using mouse, keyboard and scroll

Considering a count of activities captured versus time spent per activity, it is clear that on average, keyboard was that most common event captured at 71% followed by mouse at 19% and then scroll at 10% as depicted in Figure 4.1. On the other hand, in figure 4.2 considering time spent per activity, mouse was leading at 64%, followed by keyboard at 32% and finally scroll at 4% of the total time.

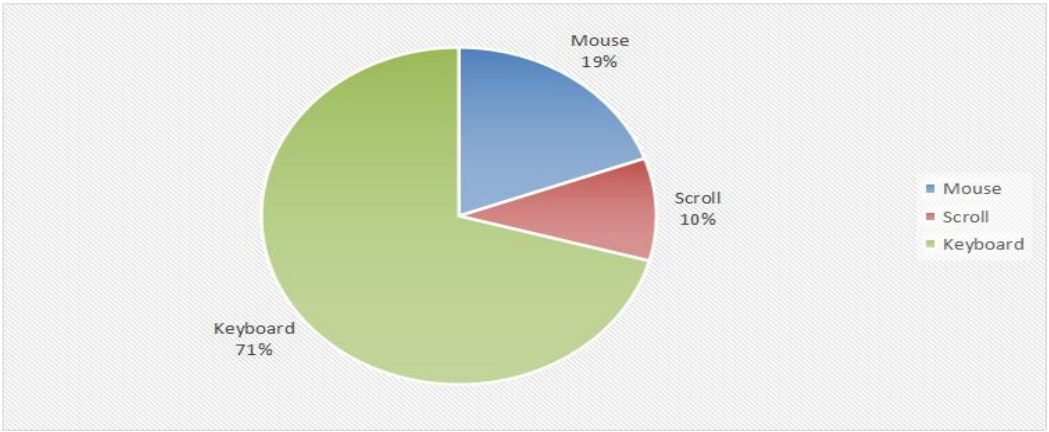

Figure 4.1 Comparison of events captured from mouse, keyboard and scroll

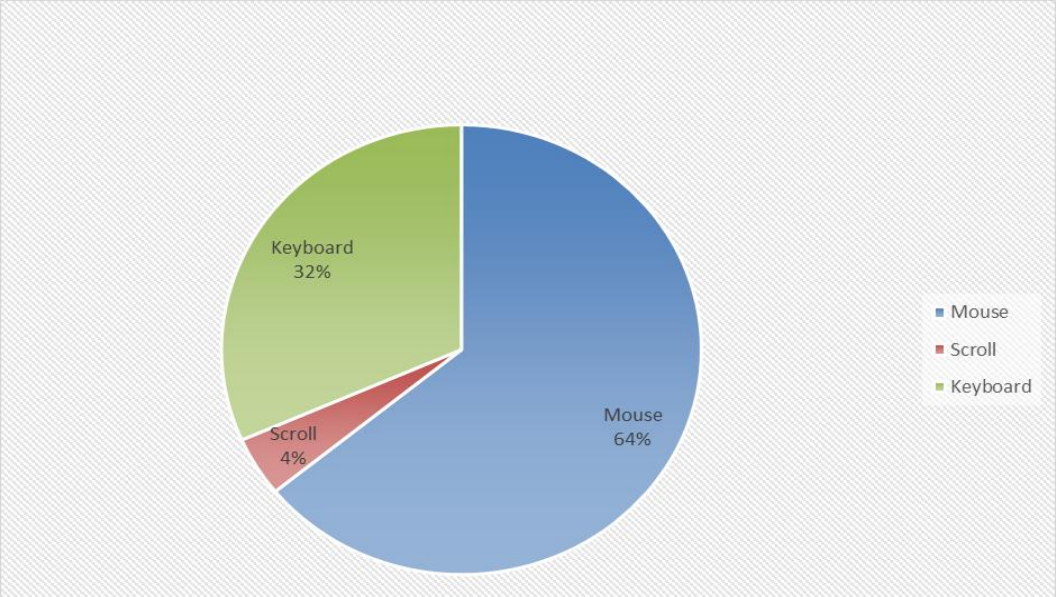

Figure 4.2: Comparison of time spent using mouse, keyboard and scroll

From the comparison above, we observe that although more keyboard activities were captured, the subjects spent more time using the keyboard see figure 4.3 and figure 4.4. This observation is true because mouse data was available throughout the session. This is because mouse data which forms our ground truth are always available regardless of whether users perform activities using the keyboard or not. Thus, in events where subjects are exposed in scenarios that require use of keyboard, we cannot rely solely of mouse as the source of focus. We can easily extract gaze region from keyboard activities by capturing the position of the input elements where users are actively pressing the keys. In some cases e.g. session 1459420025494, 1459596135462 and 1459420764271, see Appendix 1, we find out that subjects spent more time on mouse and less events were captured in comparison to keyboard. For scroll activities, we find out that subjects generated more events in shorter period of time compared to mouse and keyboard. This leads us to another important conclusion that spending more time with an input device may not necessarily mean that you are performing activities. Activities were considered to have been done when users generated events via an input device.

In this session we find out that the subject generated most activities from the keyboard and least activities from the scroll as seen in Figure 4.1. While considering time, we find out that the subject spent almost equal amount of time using keyboard and mouse and least amount of time scrolling as seen in Figure 4.2. Like we observed from the average of activities and time in the above section, generating more activities from a single input device does not necessarily mean that we can consider the input device as the source of focus.

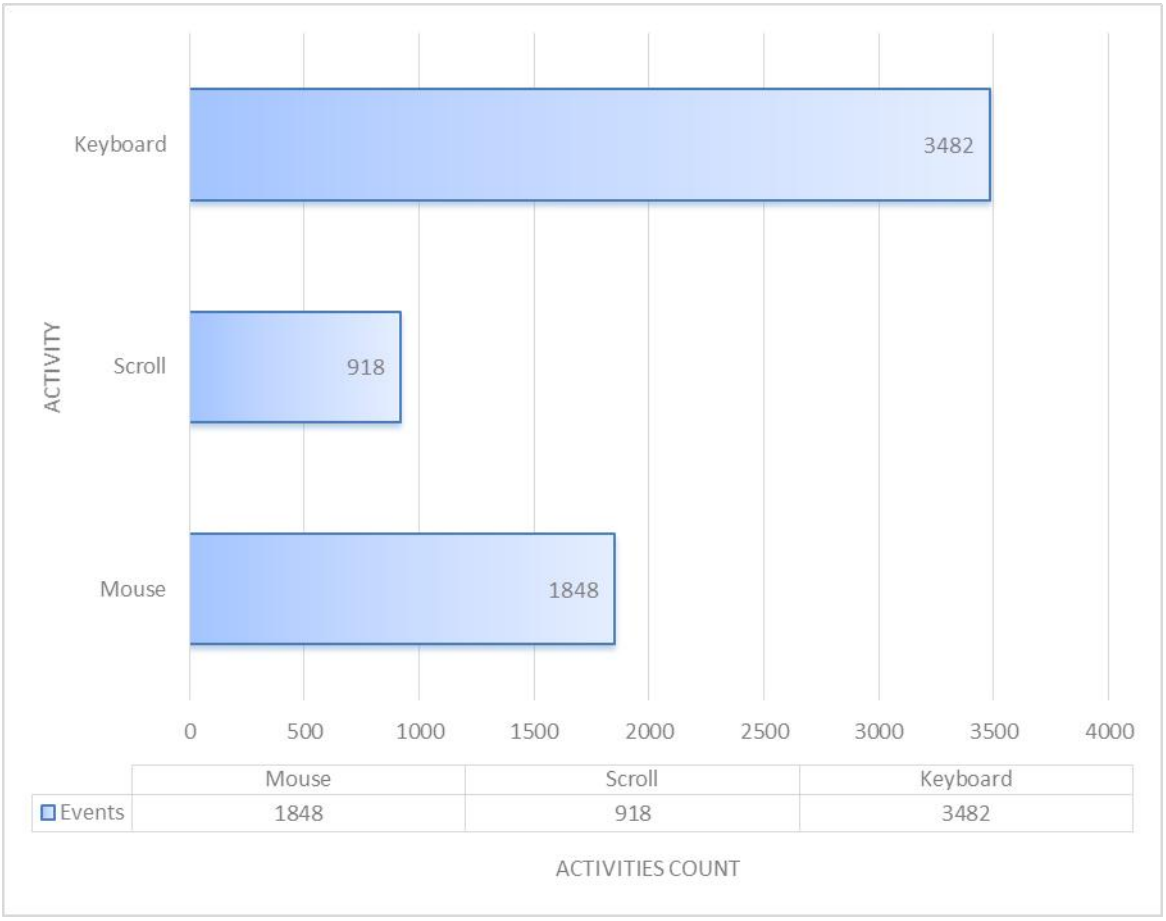

Figure 4.3: Comparison of Activities Count for mouse, keyboard and scroll

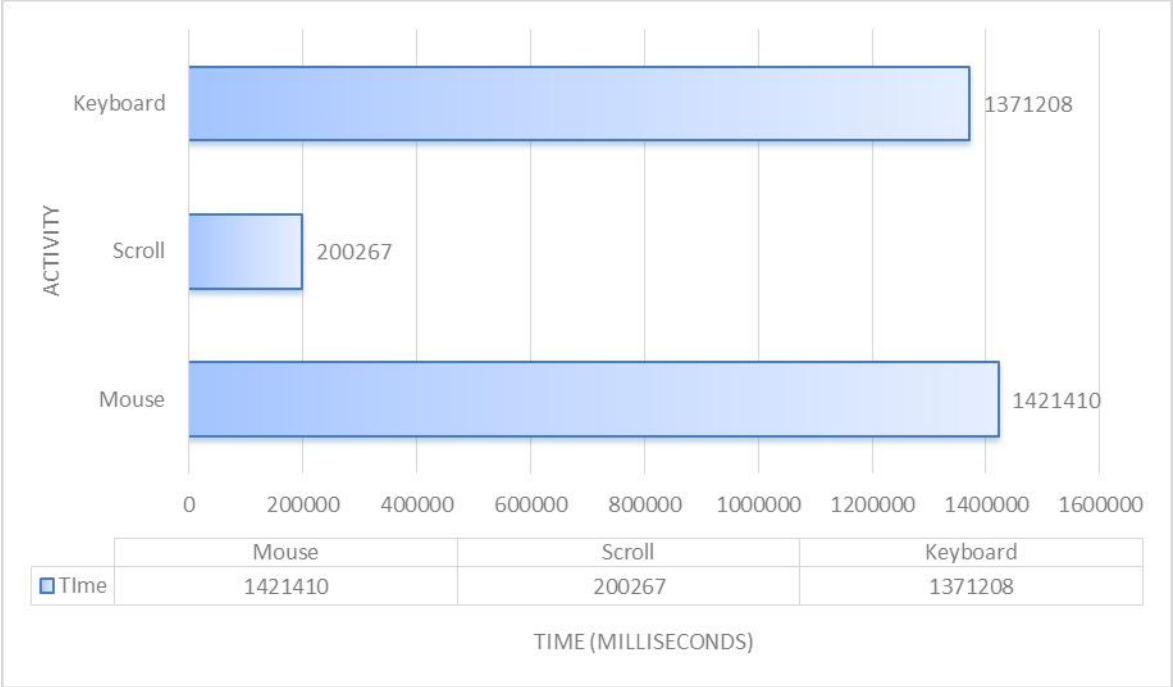

Figure 4.4: Comparison of time spent on event for mouse, keyboard and scroll

4.2 Gaze estimation

The gaze position was estimated using the co-ordinates depicting the computer screen region. The line is the mouse cursor movement path as the user interacts with the page. This is assumed in most studies to be the gaze region since it is only the mouse cursor data that is used to predict the gaze region. The developed algorithm introduced considers data from the other input, keyboard activities and scroll activities. During the entire session, the various input sources are used and their data is collected for gaze prediction. We use the LMG metric regression model in equation 6 and equation 7 to compute the weights (coefficients) for each of the three distinct features for x and y coordinates. The model normalizes positions to less and distinct version depicting lengthy periods of saliency and extrapolates to possible coordinates depicting instances of saccades. Huang *et al.* (2012) used a similar approach with the only difference being that only mouse data was used. The estimated gaze region was then visualized in the form of (x, y) coordinates.

4.3 Path followed and gaze estimated from path followed when using mouse, keyboard and scroll per session

To find out paths followed by each activity, Google charts, a powerful, simple to use, and free Java script library was used to visualize the subject's paths of action while performing activities. Raw positions are observed from randomly selected session with id **1459423656482**. Using Google Line Chart and Bar Chart, we draw charts for paths followed using mouse, keyboard and scroll in Figure 4.5, 4.7 and 4.9.

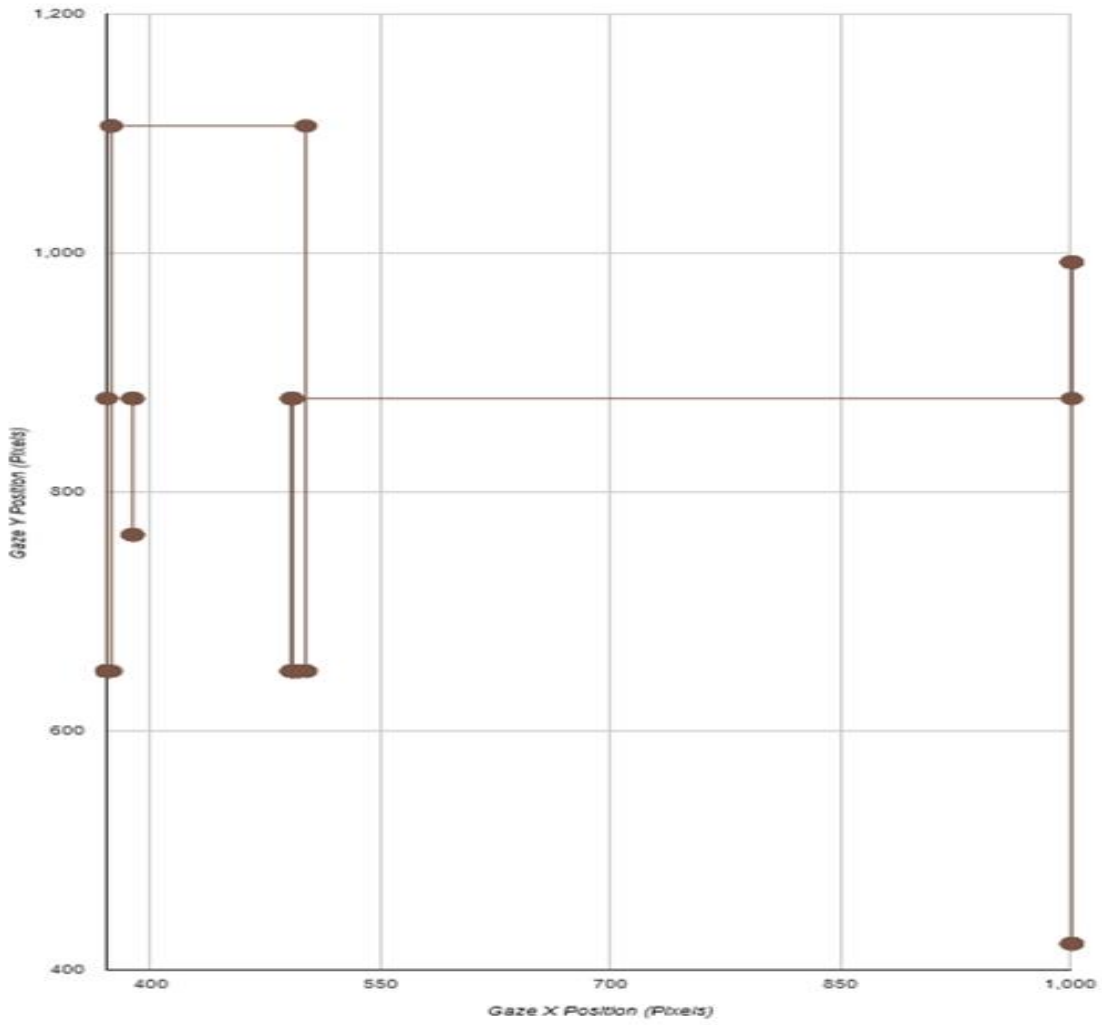

Figure 4.5: Points of focus taken from the scroll activities data

In the diagram Figure 4.1, we can see paths followed by subject during the session were square like moves. The chart appears so because mainly scroll events are mainly vertical. During analysis, for each Y position of the scroll was captured. From the theoretical framework, we hypothesized when a document scrolls vertically from position Y_1 to Y_2, we would use scroll amount from Y_1 to Y_2 to compute possible position at Y_2 in reference to mouse position at document position Y_1. Thus, the X position was picked from the current Mouse X position to achieve chart plotting. The Y position of scroll was captured when scroll event occurred. In the session, we can note from the graph of comparing activities count and event count that a considerable number of scroll events were captured. By plotting a scroll line chart with the help of mouse, we can easily estimate the gaze position of while the subject was scrolling through the document during the session. Thus scroll is a useful tool for estimating gaze position. To estimate gaze using the scroll activities, a LMG metric regression equation 6 and equation 7 are used. The model helps us achieve the following Chart.

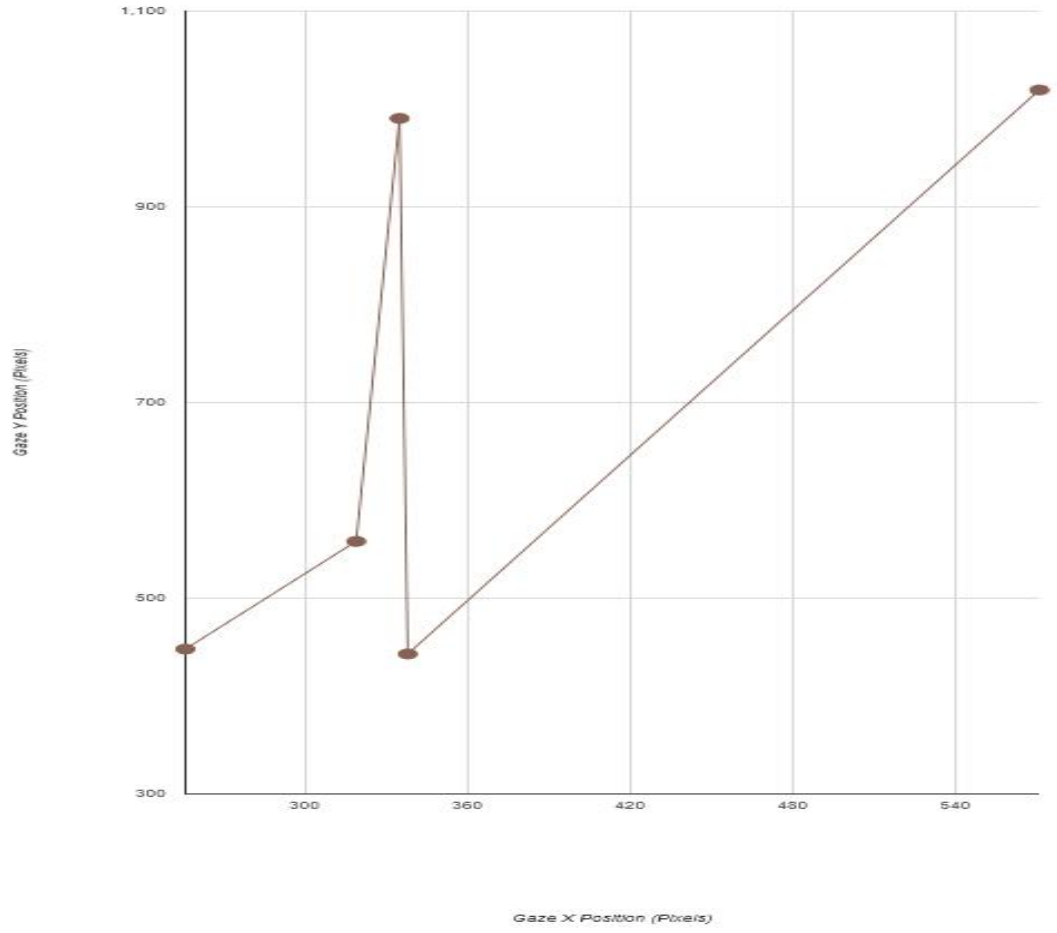

Figure 4.6: Gaze path estimated using scroll activities data

As illustrated, the estimated gaze is now smoother compared to raw scroll activities data in the previous diagram. The diagram tells us that subject spent most of the time of top and middle part of the document and in few instances of time, the scroll was done to the lower parts of the page.

While considering keyboard data, we find out that activities were mostly aligned in a horizontal manner, this is so because we use left to right language based keyboards (mostly English). The diagram below shows points of focus taken from the keyboard activities.

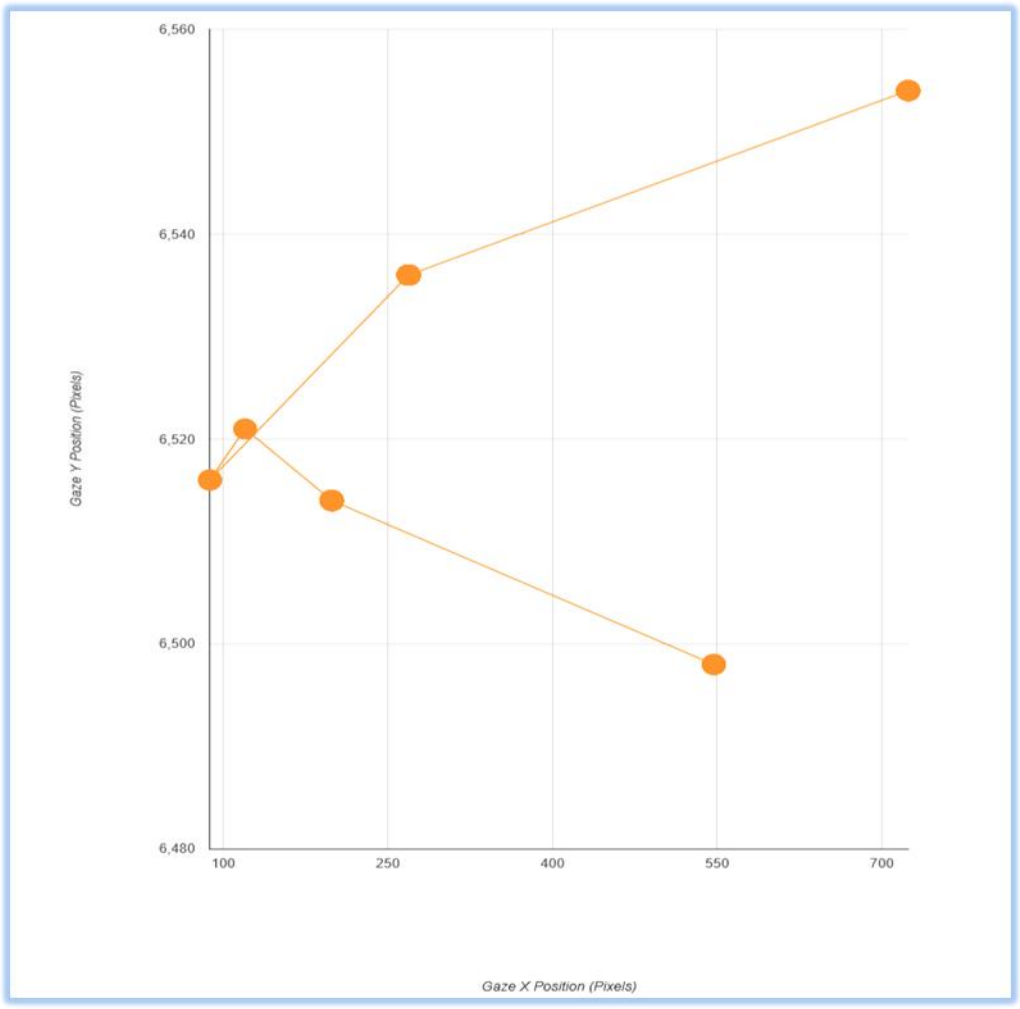

Figure 4.7: Points of focus taken from the keyboard activities

By applying LMG metric regression equation 6 and equation 7, we can achieve the following chart that shows the gaze path estimated using keyboard data.

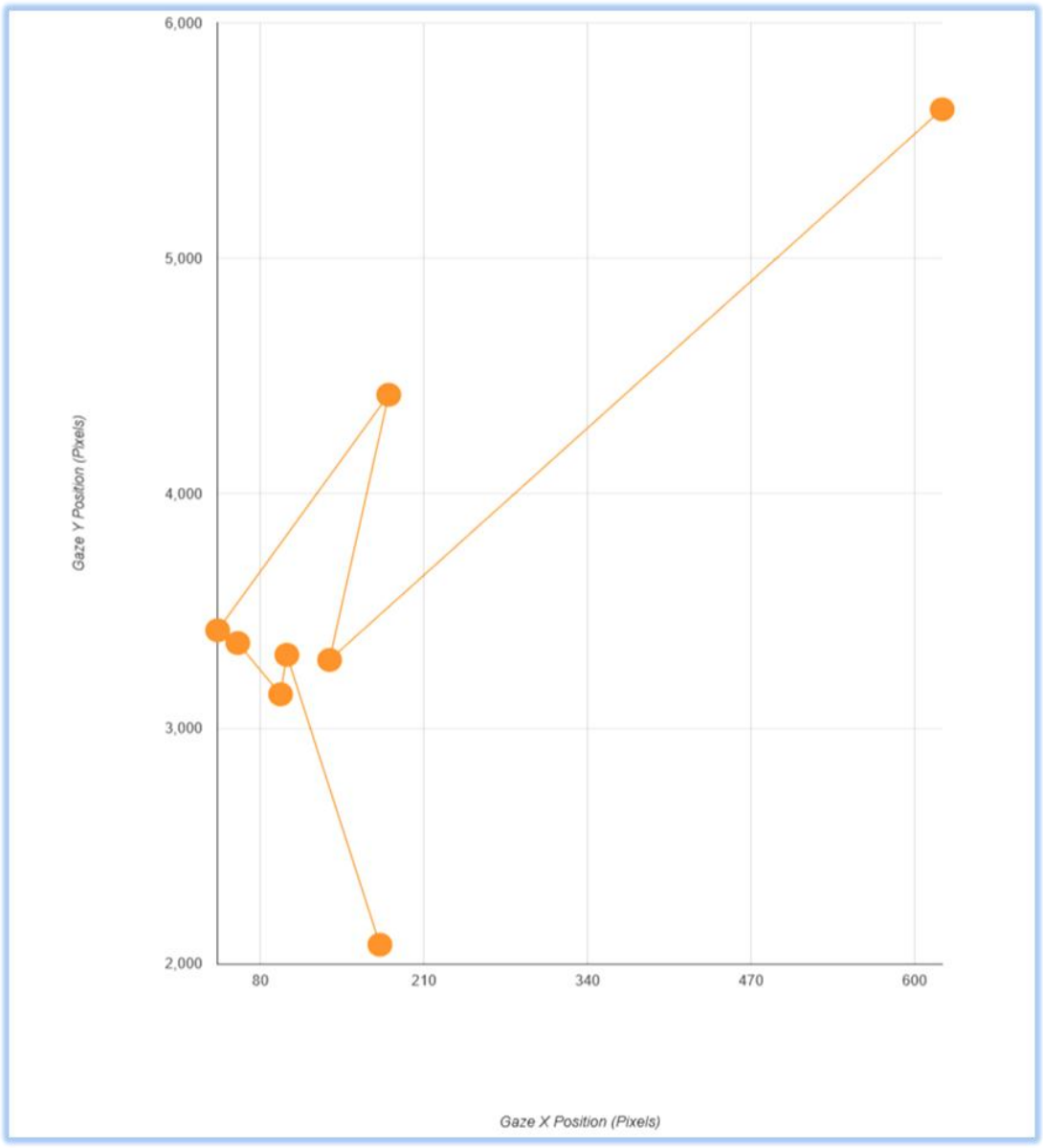

Figure 4.8: Gaze path estimated using keyboard activities data

Keyboard activities are mostly generated when users press keys to input some data, mainly data-capture forms. When subjects perform activities using the keyboard, we can capture gaze position by finding the X and Y position of the input element they are keying in text on the page document. Mainly users change gaze position from the keyboard to the input element as they input some text. Thus keyboard is an indispensable source of gaze position if subjects perform some input via the keyboard. We can therefore deduce that keyboard is an important source of gaze.

Lastly but not the lease we consider the mouse. Mouse position data is always available regardless of whether subjects use keyboard or scroll via keyboard or even mouse. Mouse position can be captured easily by capturing mouse pointer position when an activity is performed. As we can see in the following diagram, keyboard activities are forested all over and especially denser in the lower region of the pages.

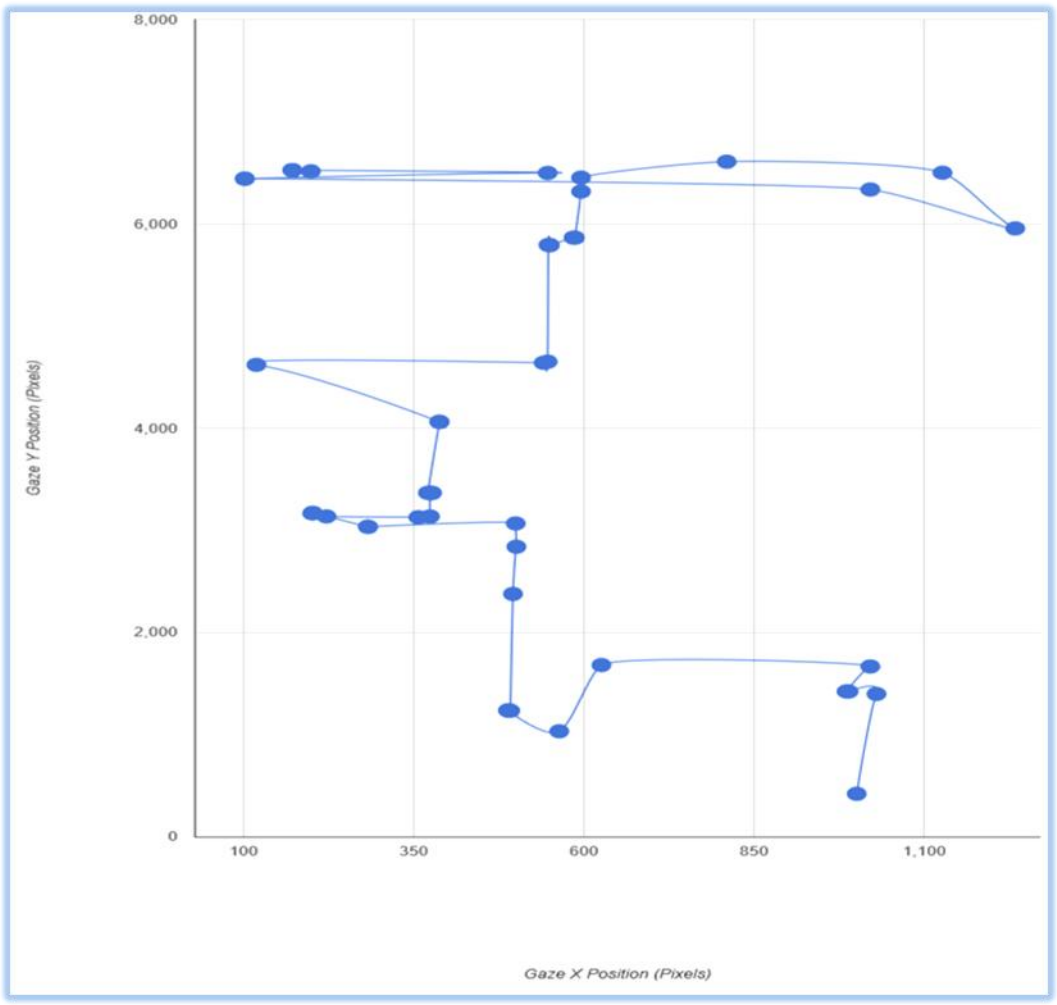

Figure 4.9: Points of focus taken from the mouse activities

We can deduce that subjects' mouse activities were concentrated in some rectangular sections of the page most likely in left-right manner. Subjects may have been reading through sentences while moving the mouse as they read. When estimating mouse path gaze, we find out that subjects gaze mostly in the middle to the lower parts of the page, see diagram below. Gaze may have started from the top-left section of the page which is typical to many users. Gaze estimation was achieved using LMG metric regression equation 6 and equation 7.

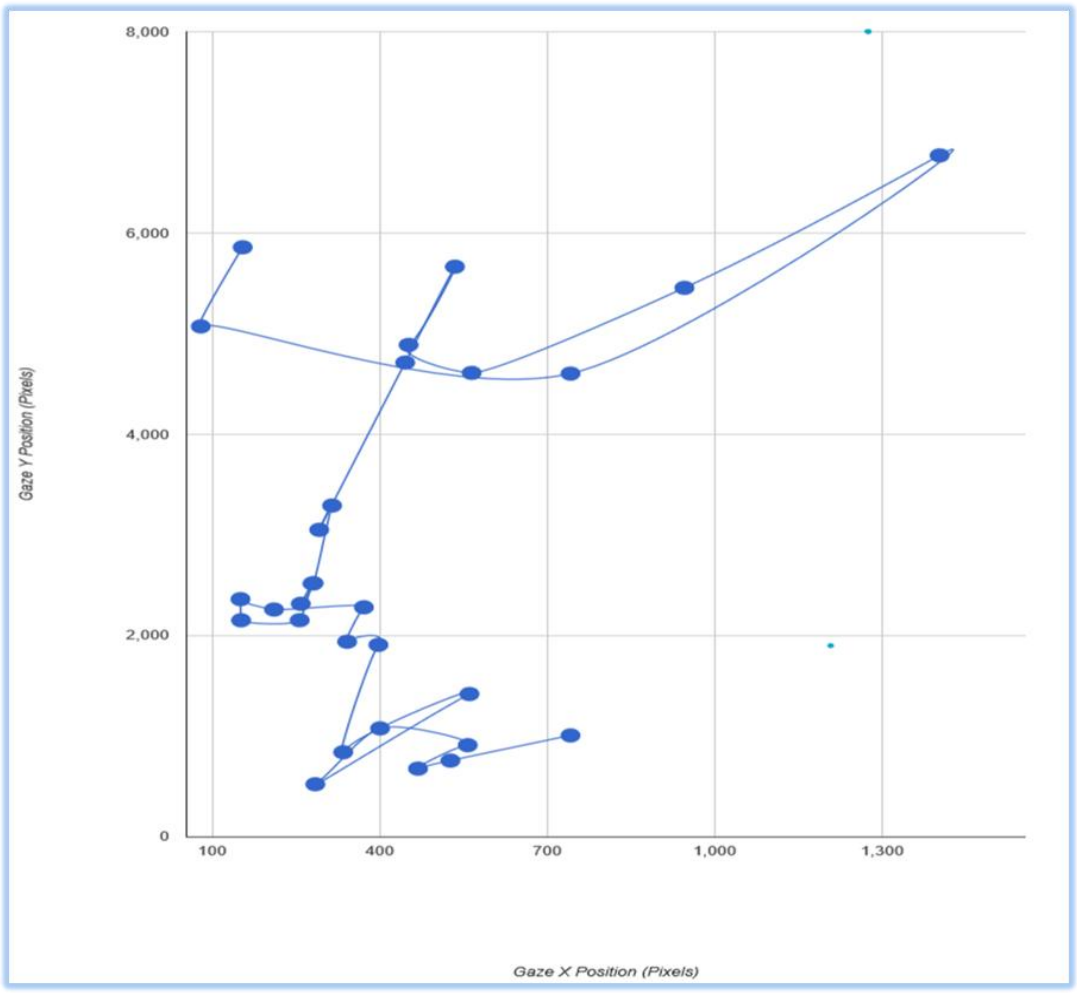

Figure 4.10: Gaze path estimated using mouse activities data

As discussed above, mouse is an important source of gaze because it is always available.

4.4 Results Validation

In this research, the problem for investigation was accurate gaze region identification on web pages using more factors that influence eye-gaze not only mouse data. Several factors including, time, mouse behaviors, document scroll patterns among others were highlighted to affect estimated gaze position. In this work, we use mouse data as ground truth for what has been done. However, we were able to collect keyboard usage data and scroll data too, as indicated in figure 4.6, 4.8 and 4.10. This gives us a clear indication that we cannot ignore keyboard and scroll and purely focus on mouse data when estimating user gaze on web pages. To clarify this, we attempt to combine the three gaze paths over time and we end up with the aggregate gaze path in Figure 4.11 and the activities timeline in figure 4.12.

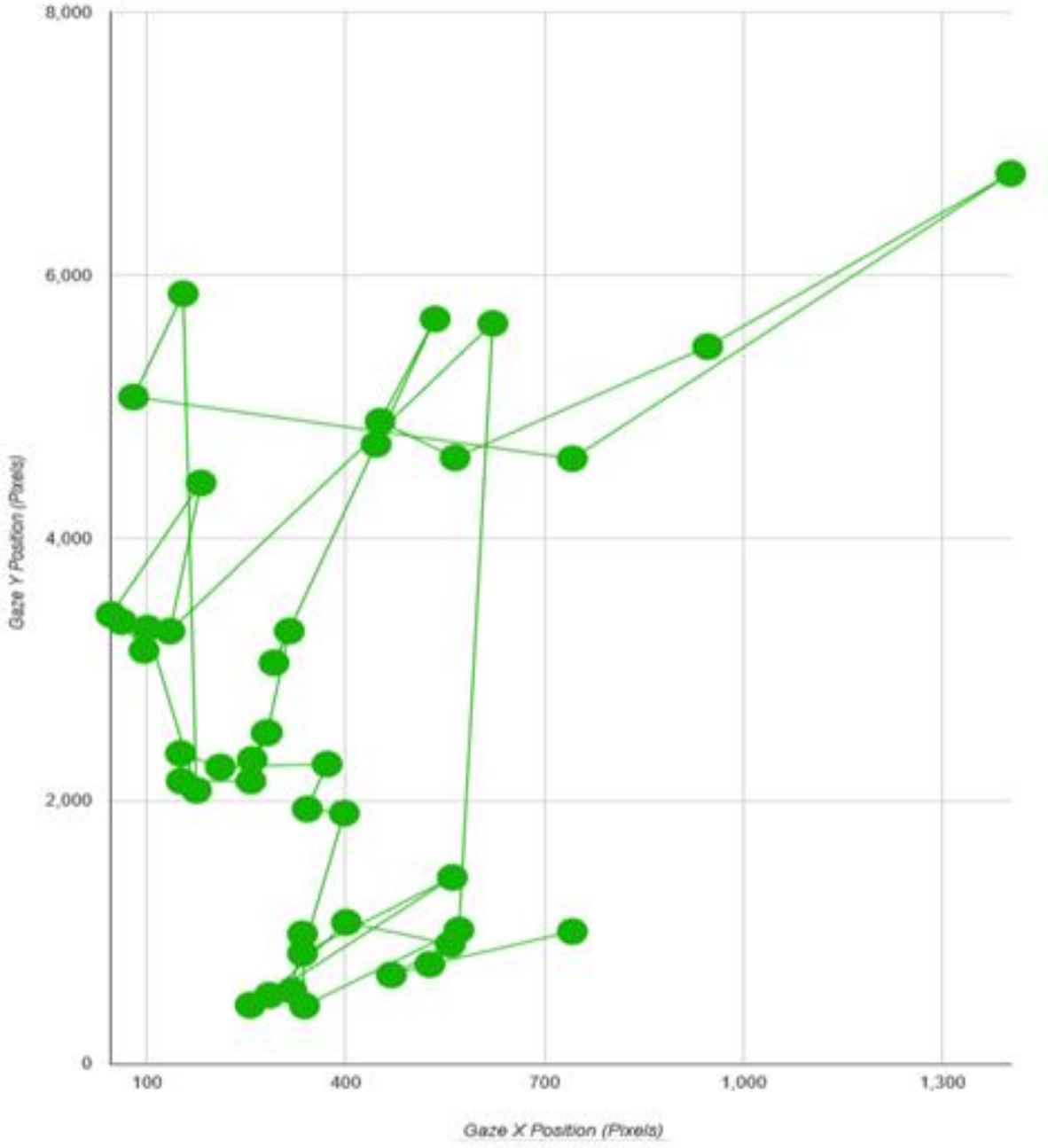

Figure 4.11: Aggregated gaze path

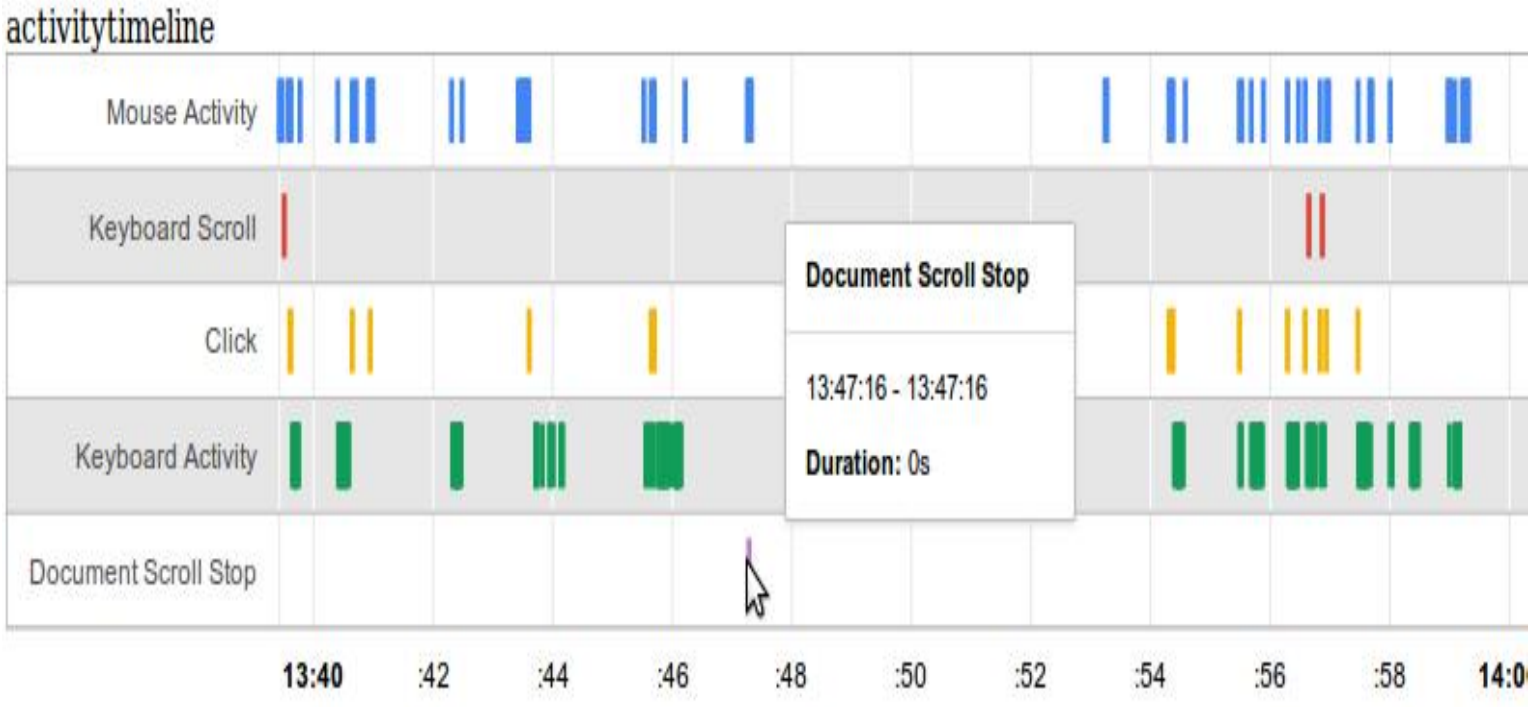

Figure 4.12: Activities Timeline

CHAPTER FIVE

SUMMARY AND CONCLUSION

5.1 Summary

In this research, the problem for investigation was how to deal with inaccurate gaze region identification on web pages using mouse data as a result of failing to consider other factors that influence eye-gaze. Several factors including, time, mouse behaviors, document scroll patterns among others were highlighted to affect estimated gaze position.

From previous research, gaze tracking systems have mainly depended on head-mounted cameras, which make them cumbersome. This limitation restricts them to a laboratory set up. Attempts to use mouse cursor data for gaze tracking in recent studies have produced inaccurate results because other factors that influence eye-gaze region have not been considered. Recent studies of identifying gaze and cursor alignment have shown a correlation but more study is needed to determine gaze fixations, especially when cursor fixations cannot be reliably interpreted as gaze regions. This is due to the fact that many operations performed by mouse can also be performed using the keyboard. Most people use the keyboard when they are not using the mouse but concurrent use of both input devices is rare. In light of this, monitoring keyboard strokes is critical in tracking user activities especially on web pages containing forms.

We undertook an exercise to determine if user activities other than using the mouse cursor affect gaze region. In previous studies, a user's gaze region was established and aligned to the mouse cursor. In the study, a web application with a set of questions was administered to a group of students. The students were required to answer the questions by typing in answers, checking radio buttons and check boxes as well as selecting options from drop down boxes.

A JavaScript code was embedded to collect usage data as students attempted the questions provided. Data collected included mouse activities such as clicks and moves, keyboard presses and scrolls in the document. The data was then pushed to a remote server via RESTful URL for storage in a SQL database. An algorithm that takes into consideration the captured user activities was developed to estimate a probable gaze region in the form of (x,y) coordinates from the data captured. The estimated gaze region was then visualized against mouse position in the form of (x, y) coordinates.

When data from keyboard activities, mouse activities and scroll activities are combined and the LMG metric regression model is used to estimate gaze path, the gaze path changes completely. This is because activities from keyboard and scroll adds additional gaze points in the path. In Figure 4.8, we can see in timeline diagram that over time, there was switching of activities among keyboard, mouse and scroll.

5.2 Conclusion

In this work, we have studied mouse clicks and drags, document scrolls, key presses and cursor movements which are the main actions may influence the gaze region over time depending on user actions when interacting with web pages and other documents. The gaze region of a person can then be computed by identifying gaze positions when an action is performed using these input devices.

We have established that, user gaze does not follow the mouse activities all the time because document scrolls and keyboard activities are evidently occurring and therefore influence the gaze region. This gives us a clear indication that we cannot ignore keyboard and scroll and purely focus on mouse data when estimating user gaze on web pages. To clarify this, we attempted to combine the three gaze paths over time.

It is hoped that this research undertaking will find wide application in user-based attention research in large scale as it does not require controlled lab settings, expensive wearable equipment as well as demand for user calibration.

5.3 Recommendations

I recommend further research to be conducted with camera-based tracking devices to verify the accuracy of the gaze estimated region. The main aim of this work was to estimate gaze region in web pages in non-laboratory based setup where users do not need to wear head mounted cameras. Since people involved in this work were using either laptop or desktop computers, further work should also be done to determine how gaze region can be estimated in hand held devices such as mobile phones and tablets. It is also important to conduct more studies on important usability requirements for eye gaze tracking devices with a view to augmenting their applicability in regards to more accurate gaze estimation.

REFERENCES

Anderson, D. R. (2007). *Fundamentals of Business Statistics.* New York: Cengage Learning.

Al-Rahayfeh, A., & Faezipour, M. (2013). Eye tracking and head movement detection: A state-of-art survey. *IEEE journal of translational engineering in health and medicine, 1*, 21- 22.

Ali, M. & Arie, V. (2007). Migrating Multi-page Web Applications to Single-page AJAX Interfaces. In Proceedings of the 11th European Conference on Software Maintenance and Reengineering (CSMR '07). Washington, DC: IEEE Computer Society.

Altaboli, A., & Lin, Y. (2011). Investigating effects of screen layout elements on interface and screen design aesthetics. *Advances in Human-Computer Interaction, 5,* 34 -54.

Booth, D. (2008) A Review of Methodologies for Analyzing Websites. *Handbook of Research on Web Log Analysis,* 143-164.

Ashraf, D. & Kamaljit I. L, (20011). The Impact of the New Web 2.0 Technologies in Communication, Development, and Revolutions of Societies, *Journal of Advances in Information Technology, 2*(4), 204-216.

Atterer, R., Wnuk, M., & Schmidt, A. (2006*). Knowing the user's every move: User activity tracking for website usability evaluation and implicit interaction*. In Proceedings of the 15th International Conference on World Wide Web (pp. 203–212). New York: ACM.

Asteriadis, S., Karpouzis, K. & Kollias, D. (2014). Visual Focus of Attention in Non-calibrated Environments using Gaze Estimation. *International Journal of Computer Vision, 107*(3), 293–316.

Balzarotti, D., Cova, M., & Vigna, G. (2008). ClearShot: Eavesdropping on Keyboard Input from Video. Proceedings of the 2008 IEEE Symposium on Security and Privacy, (SP '08), 170-183.

Bellettini, C., Marchetto, A. & Trentini, A. (2004). WebUml: reverse engineering of web applications. Conference: Proceedings of the 2004 ACM Symposium on Applied Computing (SAC) pp.1662-1669. DOI: *10.1145/967900.968231*

Blignaut, P. & Beelders, T. (2009). The effect of fixational eye movements on fixation identification with a dispersion-based fixation detection algorithm. *Journal of eye movement research, 2*(5), 4 10.

Boczon, P. (2014). State of the Art: Eye Tracking Technology and Applications. Proceedings of the 6Th Seminar on Research Trends In Media Informatics, *RTMI '14*, 8 -16.

Buscher, G., Elst, L., & Dengel, A. (2009). Segment-level display time as implicit feedback: a comparison to eye tracking. *Proceedings of SIGIR*, 67–74.

Chabane, D., Lew, S., Simovici, D., Mongy, S., & Ihaddadene, N. (2006). Eye/Gaze Tracking in Web, Image and Video Documents. Proceedings of the 14Th Annual ACM International Conference on Multimedia, *MULTIMEDIA '06,* 481-482.

Chen, o., & Lim, V. (2014). Eye Gaze and Mouse Cursor Relationship in a Debugging Task. *HCI International, 13,* 468-472.

Chennamma, H. & Yuan, X. (2013). A Survey on Eye-Gaze Tracking Techniques. *arXiv preprint arXiv*, 1312 - 6410.

Claypool, M, P. Le, M. & Wased, D. B. (2015). Implicit interest indicators. Proc. *IUI '01*, 33–40.

Cockburn, A., Quinn, P., Gutwin, C., & Fitchett, S. (2012). Improving scrolling devices with document length dependent gain. Proceedings of the SIGCHI Conference On Human Factors In Computing Systems, *CHI,'12*, 267-276.

Cognolato, M., Atzori, M., & Müller, H. (2018). Head-mounted eye gaze tracking devices: An overview of modern devices and recent advances. *Journal of Rehabilitation and Assistive Technologies Engineering, 5*, 205- 26.

Combining Gaze and Keyboard. (2015). *15th Human-Computer Interaction (INTERACT)*, Sep 2015, Germany: Bamberg.

Commissioner, P. (2014). Online Behavioral Tracking. Hong Kong: Office of the Privacy Commissioner for Personal Data. Retrieved from *http://www.pcpd.org.hk/english/publications/files/online_tracking_e.pdf*

Cooke, L. (2006). Is the Mouse a Moor Man's Eye Tracker?. Proceedings of STC. Retrieved from *https://www.clicktale.com/media/1908/cooke_mouse _eye _tracker.pdf*

Feit, A. M., Williams, S., Toledo, A., Paradiso, A., Kulkarni, H., Kane, S., & Morris, M. R. (2017, May). Toward everyday gaze input: Accuracy and precision of eye tracking and implications for design. In Proceedings of the 2017 Chi conference on human factors in computing systems (pp. 1118-1130). ACM.

Fridman,L., Langhans,P.,& Lee J.(2016). Driver Gaze Region Estimation without Use of Eye *Movement* IEEE Intelligent Systems*, 31(3),* 49 - 56.

Garaas, T. W., Nieuwenhuis, T., & Pomplun, M. (2008). A gaze-contingent paradigm for studying continuous saccadic adaptation. *Journal of neuroscience methods, 168*(2), 334-340.

Goecks, J., & Shavlik, J. (2015). Learning users' interests by unobtrusively observing their normal behavior. Proceedings of the 5Th International Conference On Intelligent User Interfaces, IUI '00, 129-132. doi:10.1145/325737.325806

Guo, Q., & Agichtein, E. (2010). Ready to buy or just browsing? detecting web searcher goals from interaction data. Proceedings of SIGIR, 130–137.

Hinckley, K., Cutrell, E., Bathiche, S., & Muss, T. (2015). Quantitative Analysis of Scrolling Techniques. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, *CHI '02,* 65-72.

Huang, J. (2011). On the Value of Page-Level Interactions in Web Search. Retrieved from http://jeffhuang.com/Final_PageLevelInteractions_HCIR11.pdf

Huang, J. (2013). Modeling user behavior and attention in search. Retrieved from: *https://digital.lib.washington.edu/researchworks/bitstream/handle/1773/24188/Huang_washington_0250E_11808.pdf?sequence=1,* June 2018

Huang, J., White, R., & Buscher, G. (2012). User see, user point: gaze and cursor alignment in web search. Proceedings of the SIGCHI Conference on Human Factors In Computing Systems, 1341-1350. Retrieved from *http://jeffhuang.com/Final_GazeCursor_CHI12.pdf,*

Israel, G. D. (2013). *Determining sample size*, Florida: University of Florida.

Iqbal, S. & Bailey, B. (2004). Using eye gaze patterns to identify user tasks. Retrieved from *https://www.interruptions.net/literature/Iqbal-GHC04.pdf,*

Isokoski, P., Joos, M., Spakov, O., & Martin, B. (2009). Gaze controlled games. *Univ Access Inf Soc, 8*(4), 323-337.

Jiang, J., He, D., & Allan, J. (2015). *Searching, browsing, and clicking in a search session: changes in user behavior by task and over time*. The 37Th

International Conference on Research and Development in Information Retrieval, 607-616.

Joachims, T., Granka, L., Pan, B., Hembrooke, H., & Gay, G. (2005). *Accurately Interpreting Clickthrough Data as Implicit Feedback.* Proceedings of The 28Th Annual International ACM SIGIR Conference On Research And Development In Information Retrieval, 154-161. Doi:*10.1145/1076034.1076063*

Kaupins, G., & Minch, R. (2006). Legal and Ethical Implications of Employee Location Monitoring. *International Journal of Technology and Human Interaction, 2*(3), 16-35.

Ken, Y., & Christof, L., (2016).Gerald Weber. Eyes and Keys: An Evaluation of Click Alternatives Retrieved from *https://pdfs.semanticscholar.org/a426/7c0c 503ed6892a2e3ef373f87b9023d243a4.pdf,*

Kohavi, R., Messner, D., Eliot, S., Lavista, J., Henne, R., Kannappan, V., & Wang, J. (2010). Tracking Users' Clicks and Submits: Tradeoffs between User Experience and Data Loss. Microsoft White Paper. Retrieved from *https://pdfs.semanticscholar.org/a426/7c0c503ed6892a2e3ef373f87b9023d243a4.pdf,*

Krejtz, K., Biele, C., Chrzastowski, D., Kopacz, A., Niedzielska, A., Toczyski, P., & Duchowski, A. (2014). Gaze-controlled gaming: immersive and difficult but not cognitively overloading. Proceedings of the 2014 ACM International Joint Conference On Pervasive And Ubiquitous Computing: Adjunct Publication, UbiComp '14 Adjunct, 1123-1129.

Kumar, M., Paepcke, A., & Winograd, T. (2007). EyePoint: practical pointing and selection using gaze and keyboard. Proceedings of the SIGCHI Conference on Human Factors In Computing Systems, *CHI '07*, 421-430.

Retrieved from *https://hci.stanford.edu/publications/2007/chi253-kumar.pdf,* June 2018

Liebling, D., & Dumais, S. (2014). Gaze and Mouse Coordination in Everyday Work. Proceedings of the 2014 ACM International Joint Conference on Pervasive And Ubiquitous Computing: Adjunct Publication, *UbiComp '14*(Adjunct), 1141-1150.

Lupu, R., Stan, A., & Ungureanu, F. (2014). Wireless Device for Patient Monitoring. Proceedings of the World Congress on Engineering 2008 Vol II WCE 2008. Retrieved from *http://www.iaeng.org/publication/WCE2008/WCE2008_pp1687-1691.pdf,*

Mansour JK & Flowe HD (2010) Eye tracking and eyewitness memory. Forensic Update, No. 101. n.p. Retrieved from: *https://dspace.lboro.ac.uk/dspace-jspui/handle/2134/20322.*

Muñoz, J., Yannakakis, G., Mulve, F., Hansen, D., Gutiérrez, G., & Sanchís, A. (2011). Towards Gaze-Controlled Platform Games. Computational Intelligence and Games *(CIG),* 47 -54.

Navalpakkam, V., Jentzsch, L., Sayres, R., Ravi, S., Ahmed, A. and Smola, A. (2013). Measurement and modeling of eye-mouse behavior in the presence of nonlinear page layouts. Proceedings of the 22nd international conference on World Wide Web pp.953-964.

Ooms, K., Coltekin, A., De Maeyer, P., Dupont, L., Fabrikant, S., Incoul, A., ... & Van der Haegen, L. (2015). Combining user logging with eye tracking for interactive and dynamic applications. *Behavior research methods, 47*(4), 977-993.

Pan, B., Hembrooke, H., Gay, G., Granka, L., Feusner, M., & Newman, J. (2004). The Determinants of Web Page Viewing Behavior: An Eye-Tracking

Study. Proceedings of the 2004 Symposium on eye Tracking Research & Applications, *ETRA '04*, 147-154.

Pan, B., Hembrooke, H., Joachims, T., Lorigo, L., Gay, G., & Granka, L. (2007). In Google We Trust: Users' Decisions on Rank, Position, and Relevance. *Journal of Computer-Mediated Communication, 12*(3), 801-823.

Peter, D. (2008). *Engineering Adaptive Web Applications: A Domain Engineering Framework*, Saarbrücken, Germany: VDM Verlag.

Piwowarski, B. (2009). Mining user web search activity with layered bayesian networks or how to capture a click in its context. Proceedings of the Second ACM International Conference On Web Search And Data Mining, *(WSDM '09),* 162-171.

Prasov, Z., Chai, J., & Jeong, H. (2007). Eye gaze in attention prediction in multimodal human machine conversation. Spring Symposium On Interaction Challenges For Artificial Assistants, Proceedings of the AAAI (March 2007). Retrieved from *http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.509.7449&rep=rep1&type=pdf,* June 2018

Reis, J. L., & Carvalho, J. Á. (2012, June). Personalized information systems: enabling technologies and architecture. In 7th Iberian Conference on Information Systems and Technologies (CISTI 2012) (pp. 1-5). IEEE.

ROBU, A. (2013). Assessment of E-Commerce Websites through a Mouse Tracking Investigation. *Review of Business and Economic Studies*, *6*(1), 107-119.

Schmucker, N. (2011). Web Tracking. SNET2 Seminar Paper – Summer Term 2011 Niklas Schmucker. Retrieved from *http://www.snet.tu-berlin.de/fileadmin/fg220/courses/SS11/snet-project/web-tracking_schmuecker.pdf,*

Schneider, F., Agarwal, S., Alpcan, T. & Feldmann, A. (2008). The new web: Characterizing Ajax Traffic. *Springer*, 31-40.

Shuhong, L., & Qiaorong, Z. (2014). A Computational Model for Object-based Visual Attention. *Research Journal of Applied Sciences*, *Engineering and Technology*, *7*(1), 42-48.

Stiefelhagen, R., Yang, J., & Waibel, A. (1997). Tracking eyes and monitoring eye gaze. In Workshop on Perceptual User Interfaces, 98-100. Retrieved from *http://www.cs.cmu.edu/~stiefel/papers/PUI97-rainer.pdf.gz,* June 2018

Villa, R., & Chalmers, M. (2001). A framework for implicitly tracking data. Proceedings of the Second DELOS Network Of Excellence Workshop on Personalization And Recommender Systems in Digital Libraries, (June 2001), 18 – 20. Retrieved from *https://www.ercim. eu/ publication/ws-proceedings/DelNoe02/ RobertVilla.pdf,*

Yerby, J. (2013). Legal and ethical issues of employee monitoring. *Online Journal of Applied Knowledge Management, 1*(2).

Yokoyama, T., Sakai, H., Noguchi, Y. and Kita, S. (2014). Perception of Direct Gaze Does Not Require Focus of Attention. *Scientific reports, 4*(3858), 1038 - 3858

Zhai, S., Smith, B., & Selker, T. (2007). Improving Browsing Performance: A study of four input devices for scrolling and pointing tasks. Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction, *INTERACT '97*, 286-293.

Zhang, Y., Bulling, A., & Gellersen, H. (2013). SideWays: a gaze interface for spontaneous interaction with situated displays**.** Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 851-860.

ZHU, J., & YANG, J. (2015). Subpixel Eye Gaze Tracking. Proceedings of the 5th IEEE International Conference on Automatic Face and Gesture Recognition, FGR (02). Retrieved from: *https://pdfs.semanticscholar.org/6aef/82d6f5f2f34ae0fa1fada99ed2f5c537fb15.pdf,*

APPENDICES

Appendix I: Table of sessions randomly selected for analysis

Session ID	Event Count per Session			Time Count per Session (Seconds)		
	Mouse	Scroll	Keyboard	Mouse	Scroll	Keyboard
1459452537665	472	102	2305	3136679	66428	1098602
1459419888554	486	127	1606	3130062	84960	961378
1459522075405	214	83	605	170753	25969	193585
1459484184176	831	151	5121	1384524	96282	1988634
1459521483245	293	174	345	2263782	91792	141500
1459517689455	1848	918	3482	1421410	200267	1371208
1459450020844	153	109	717	194438	97960	463079
1459512987162	528	463	4109	2371720	41233	1316257
1459596135462	317	70	2579	938243	18326	906672
1459436480851	631	359	4488	829991	114902	1909077
1459419932746	412	65	334	1933577	32618	184913
1459537319592	490	113	501	1576318	84723	232273
1459444367435	143	158	781	530543	8991	244090
1459497064892	218	104	2304	1055813	9868	971476
1459487563577	567	194	5032	3658076	116361	1939085
1459502725352	238	113	563	504115	54281	208121
1459494171285	1106	938	4525	3072601	564144	3177995
1459439589793	654	138	5063	1990603	25892	1383437
1459518299946	457	459	1108	787940	81100	443736
1459487046158	956	312	3975	5990366	143303	1934220
1459502818257	196	46	562	1295087	26885	267031
1459502892704	238	627	1955	1808008	62096	1070501
1459579175915	332	216	1012	1246254	42761	334530
1459498257356	228	118	605	267929	47215	267929
1459422442937	154	264	315	1624970	25436	111006
1459418795429	357	325	1721	799337	375281	1417600
1459802806853	360	115	376	690440	67698	210955
1459425366273	33	34	1972	5088	594068	2463069
1459516379880	435	391	1744	3694984	100848	732267
1459418443584	343	191	1424	4001367	178006	870170
1459492428634	1082	368	841	585512	145482	426719

1459423514469	48	47	1113	6463	15420	700037
1459418287684	186	59	1436	1408952	35088	789861
1458564888735	292	326	373	2050361	148352	239874
1459511711775	1172	145	74	275852	68263	25161
1459421682510	164	767	87	806236	23981	392704
1459500018191	330	92	830	1905900	197425	436472
1459492242018	1246	112	2958	3546852	53628	1334750
1459420764271	173	29	420	1908495	1669	188499
1459499634891	333	95	768	759820	51720	1009939
1459509856950	141	55	341	193068	10159	102003
1459514814210	152	111	668	835860	71812	421781
1459420025494	258	35	1591	2865289	29031	1101457
1459768093450	222	117	431	916020	117238	202202
1459489884228	223	121	1447	290752	76607	1443891
1459507645411	434	689	554	2113053	196730	147455
1459423656482	107	83	790	448416	18133	380592
1459436496548	1253	257	5343	5451224	253912	2406537
1459533496058	132	202	188	1767458	136391	100309
1459796449873	1032	280	918	3205372	414785	549618
Average Events				Average Time		
Mouse	453.4			Mouse	1674319.4	
Scroll	229.34			Scroll	111818.20	
Keyboard	1648			Keyboard	824285.14	

Appendix II: Table of Data for session with id 1459517689455

elementt	Scroll delta	Scroll top	event	activity	mouseX	mouseY	sessionkey	trackingtime
0	0	0	document_scroll_stop	scroll_activity	0	0	1459517689455	1459517689730
0	0	0	mouse_activity	mouse_activity	760	712	1459517689455	1459517690033
879	0	0	click	mouse_activity	269	889	1459517689455	1459517707642
879	0	0	keboard_activity	keboard_activity	269	889	1459517689455	1459517709010
879	0	0	keboard_activity	keboard_activity	269	889	1459517689455	1459517709398
879	0	0	keboard_activity	keboard_activity	269	889	1459517689455	1459517709713
879	0	0	keboard_activity	keboard_activity	269	889	1459517689455	1459517710134
879	0	0	keboard_activity	keboard_activity	269	889	1459517689455	1459517712989
879	0	0	keboard_activity	keboard_activity	269	889	1459517689455	1459517713129
879	0	0	keboard_activity	keboard_activity	269	889	1459517689455	1459517713338
879	0	0	keboard_activity	keboard_activity	269	889	1459517689455	1459517713514
879	0	0	keboard_activity	keboard_activity	269	889	1459517689455	1459517713711
879	0	0	keboard_activity	keboard_activity	269	889	1459517689455	1459517713845
879	0	0	keboard_activity	keboard_activity	269	889	1459517689455	1459517714560
879	0	0	keboard_activity	keboard_activity	269	889	1459517689455	1459517714778

Data (15 out of 6,248) for session with id 1459517689455

Appendix III: Sample Codes for Gaze Classification

```
TrackingDataSource s = findAllActivityData();
TrackingData d = new TrackingData();
String activity = s.getEvent().trim(),a="";
if(activity.equals("click")||activity.equals("contextmenu")||activity.equals("copy")||activit
y.equals("cut")||activity.equals("dblclick")||activity.equals("drag")||activity.equals("paste
")||activity.equals("select")){
        d.setActivity("mouse_activity");
}
else
if(activity.equals("mouse_scroll")||activity.equals("keyboard_scroll")||activity.equals("elem
_scroll_stop")||activity.equals("document_scroll")||activity.equals("document_scroll_stop")){
        d.setActivity("scroll_activity");
}
else{
        d.setActivity("keyboard_activity");
}
```

Appendix IV: Sample Codes for Gaze Estimation

```
public List<model.GazeXY> findMouseActivityBySessionkey(Long sessionkey) {

List data=trackerDao.findMouseActivityBySessionkey(sessionkey);
SimpleDateFormat localDateFormat = new SimpleDateFormat("HH:mm:ss");
                List<model.GazeXY> mouseLocationBuffer =new ArrayList<model.GazeXY>();
                DateTime[] trackingTimeBuffer =null;
                //While useractivitydata is not null
                if(tdata!=null){
                        trackingTimeBuffer = new DateTime[ tdata.size()];
                        int count = 0;
                        for(model.Tracker d : tdata)
                        {
                                model.GazeXY mouseLocation= new model.GazeXY();
                                mouseLocation.setX(d.getMouseX());
                                mouseLocation.setY(d.getMouseY());
                                mouseLocation.setTrackingTime(new
DateTime(d.getTrackingtime()));
                                if(trackingTimeBuffer[0]==null){
                                        mouseLocationBuffer.add(mouseLocation);
                                }
                                else{
if(trackingTimeBuffer[count].getMillis()-
mouseLocationBuffer.get(count).getTrackingTime().getMillis()>100) {
                                                mouseLocationBuffer.add(mouseLocation);
                                        }
                                }
                                trackingTimeBuffer[count]                =                new
DateTime(d.getTrackingtime());
                        }
                }
                return mouseLocationBuffer;
        }

        public List<model.GazeXY> findKeyboardActivityBySessionkey(Long sessionkey) {
                List tdata=trackerDao.findKeyboardActivityBySessionkey(sessionkey);
                SimpleDateFormat localDateFormat = new SimpleDateFormat("HH:mm:ss");
                List<model.GazeXY> keyboardLocationBuffer =new ArrayList<model.GazeXY>();
                DateTime[] trackingTimeBuffer =null;
                //While useractivitydata is not null
                if(tdata!=null){
                        trackingTimeBuffer = new DateTime[ tdata.size()];
                        int count = 0;
                        for(model.Tracker d : tdata)
                        {
                                model.GazeXY caretLocation= new model.GazeXY();
                                caretLocation.setX(d.getMouseX());
                                caretLocation.setY(d.getMouseY());
                                caretLocation.setTrackingTime(new
DateTime(d.getTrackingtime()));
                                if(trackingTimeBuffer[0]==null){
                                        keyboardLocationBuffer.add(caretLocation);
                                }
                                else{
if(trackingTimeBuffer[count].getMillis()-
keyboardLocationBuffer.get(count).getTrackingTime().getMillis()>100){
                                        keyboardLocationBuffer.add(caretLocation);
                                        }
                                }
                                trackingTimeBuffer[count]                =                new
DateTime(d.getTrackingtime());
                        }
                }
                return keyboardLocationBuffer;
```

```
        }
public List<model.GazeXY> findScrollActivityBySessionkey(Long sessionkey) {
                List tdata=trackerDao.findScrollActivityBySessionkey(sessionkey);
                //SimpleDateFormat localDateFormat = new SimpleDateFormat("HH:mm:ss");
                List<model.GazeXY> scrollLocationBuffer =new ArrayList<model.GazeXY>();
                DateTime[] trackingTimeBuffer =null;

                if(tdata==null || tdata.size()==0){
                        return scrollLocationBuffer;
                }
                List                            lastActivityInSession                           =
trackerDao.findLastNonScrollActivityInSessionBeforeTime(sessionkey,tdata.get(0).getTrackingti
me());
                long lastScrollTopInBuffer = 0;
                if(lastActivityInSession!=null && lastActivityInSession.size()>0){
                        lastScrollTopInBuffer = lastActivityInSession.get(0).getMouseY();
                }
                if(tdata!=null){
                        trackingTimeBuffer = new DateTime[ tdata.size()];
                        long lastScrollDelta = 0;
                        int count = 0;
                        for(model.Tracker d : tdata)
                        {
                                model.GazeXY scrollLocation = new model.GazeXY();
                                long scrollTop = d.getDocscrolltop();
                                scrollLocation.setTrackingTime(new
DateTime(d.getTrackingtime()));
                                scrollLocation.setX(d.getMouseX());//X Will not change

                                if(d.getDocscrolltop()!=0){
                                        if(d.getDocscrolltop()>lastScrollTopInBuffer){
                                                scrollLocation.setY((int)
(lastScrollTopInBuffer + (d.getDocscrolltop()-lastScrollTopInBuffer)));
                                        }
                                        else{
                                                scrollLocation.setY((int)
(lastScrollTopInBuffer - (lastScrollTopInBuffer-d.getDocscrolltop())));
                                        }
                                }
                                else if(d.getElemscrolltop()>0){
                                        if(d.getElemscrolltop()>lastScrollTopInBuffer){
                                                scrollLocation.setY((int)
(lastScrollTopInBuffer + (d.getElemscrolltop()-lastScrollTopInBuffer)));
                                        }
                                        else{
                                                scrollLocation.setY((int)
(lastScrollTopInBuffer - (lastScrollTopInBuffer-d.getElemscrolltop())));
                                        }
                                }
                                else {
                                        long     yPos     =     d.getDocscrolldelta()     +
lastScrollTopInBuffer;
                                        scrollLocation.setY((yPos>=0)?yPos:0);
                                }
                                scrollLocationBuffer.add(scrollLocation);
                                lastScrollDelta += d.getDocscrolldelta();

                                trackingTimeBuffer[count]                =                new
DateTime(d.getTrackingtime());
                        }
                }
                System.err.println(scrollLocationBuffer);
                return scrollLocationBuffer;
        }
public showKeyboardGaze(Long sessionkey) {
List<model.GazeXY> keyboardLocationBuffer = findKeyboardActivityBySessionkey(sessionkey);
        List<model.GazeXY> listgx =new ArrayList<model.GazeXY>();
        long startTime = keyboardLocationBuffer.get(0).getTrackingTime().getMillis();
        long dwellTime = keyboardLocationBuffer.get(0).getTrackingTime().getMillis();
```

```
        long lastActivityTime = 0;
        model.GazeXY lastgaze = new model.GazeXY();
        for(model.GazeXY d : keyboardLocationBuffer)
        {
                model.GazeXY g = new model.GazeXY();
                if(lastActivityTime>0           &&           d.getTrackingTime().getMillis()-
lastActivityTime>0){
        if(d.getTrackingTime().getMillis()-lastgaze.getTrackingTime().getMillis()>=1000){
                dwellTime=(d.getTrackingTime().getMillis()-startTime)/1000;
                long tm = (d.getTrackingTime().getMillis()-lastActivityTime)/1000;
                long td = dwellTime;
                long cx = d.getX();
                long cy = d.getY();
                long  currentX=(long)  (cx  +  Math.log10(td)  +  Math.log10(tm)  +  (cx  *
Math.log10(td)) + (cx * Math.log10(tm)));
                currentX*=2;
                long  currentY=(long)  (cy  +  Math.log10(td)  +  Math.log10(tm)+  cy  *
Math.log10(td) + cy * Math.log10(tm));
                currentY*=2;
                g.setX(d.getX());
                g.setY(d.getY());
                g.setGazeX(currentX/10);
                g.setGazeY(currentY/10);
                g.setTrackingTime(d.getTrackingTime());
                lastgaze.setGazeX(currentX/10);
                lastgaze.setGazeY(currentY/10);
                lastgaze.setTrackingTime(d.getTrackingTime());
                System.out.println(g);
                listgx.add(g);
             }

                lastgaze.setGazeX(d.getX());
                lastgaze.setGazeY(d.getY());
                lastgaze.setTrackingTime(d.getTrackingTime());
                lastActivityTime = d.getTrackingTime().getMillis();
        }
        return listgx;
}
public List<model.GazeXY> showKeyboardGaze(Long sessionkey) {
List keyboardLocationBuffer = findKeyboardActivityBySessionkey(sessionkey);
        List listgx =new ArrayList<model.GazeXY>();
        long startTime = keyboardLocationBuffer.get(0).getTrackingTime().getMillis();
        long dwellTime = keyboardLocationBuffer.get(0).getTrackingTime().getMillis();
        long lastActivityTime = 0;
        model.GazeXY lastgaze = new model.GazeXY();
        for(model.GazeXY d : keyboardLocationBuffer)
        {
                model.GazeXY g = new model.GazeXY();
            if(lastActivityTime>0 && d.getTrackingTime().getMillis()-lastActivityTime>0){
                        System.out.println(d);
                        if(d.getTrackingTime().getMillis()-
lastgaze.getTrackingTime().getMillis()>=1000){
                                dwellTime=(d.getTrackingTime().getMillis()-startTime)/1000;
                                long       tm       =       (d.getTrackingTime().getMillis()-
lastActivityTime)/1000;
                                long td = dwellTime;
                                long cx = d.getX();
                                long cy = d.getY();
                                long currentX=(long) (cx + Math.log10(td) + Math.log10(tm) +
(cx * Math.log10(td)) + (cx * Math.log10(tm)));
                                currentX*=2;
                                long currentY=(long) (cy + Math.log10(td) + Math.log10(tm)+
cy * Math.log10(td) + cy * Math.log10(tm));
                                currentY*=2;
                                g.setX(d.getX());
                                g.setY(d.getY());
                                g.setGazeX(currentX/10);
                                g.setGazeY(currentY/10);
                                g.setTrackingTime(d.getTrackingTime());
```

```
                                lastgaze.setGazeX(currentX/10);
                                lastgaze.setGazeY(currentY/10);
                                lastgaze.setTrackingTime(d.getTrackingTime());
                                System.out.println(g);
                                listgx.add(g);
                        }
}
                lastgaze.setGazeX(d.getX());
                lastgaze.setGazeY(d.getY());
                lastgaze.setTrackingTime(d.getTrackingTime());
                lastActivityTime = d.getTrackingTime().getMillis();
        }
        return listgx;
}
public List<model.GazeXY> showScrollGaze(Long sessionkey) {
        List<model.GazeXY> scrollLocationBuffer = findScrollActivityBySessionkey(sessionkey);
        List<model.GazeXY> listgx =new ArrayList<model.GazeXY>();
        long startTime = scrollLocationBuffer.get(0).getTrackingTime().getMillis();
        long dwellTime = scrollLocationBuffer.get(0).getTrackingTime().getMillis();
        long lastActivityTime = 0;
        model.GazeXY lastgaze = new model.GazeXY();
        for(model.GazeXY d : scrollLocationBuffer)
        {
                model.GazeXY g = new model.GazeXY();
                if(lastActivityTime>0           &&           d.getTrackingTime().getMillis()-
lastActivityTime>0){
                        if(d.getTrackingTime().getMillis()-
lastgaze.getTrackingTime().getMillis()>=1000){
                                dwellTime=(d.getTrackingTime().getMillis()-startTime)/1000;
                                long       tm       =       (d.getTrackingTime().getMillis()-
lastActivityTime)/1000;
                                long td = dwellTime;
                                long cx = d.getX();
                                long cy = d.getY();
                                long currentX=(long) (cx + Math.log10(td) + Math.log10(tm) +
(cx * Math.log10(td)) + (cx * Math.log10(tm)));
                                currentX*=2;
                                long currentY=(long) (cy + Math.log10(td) + Math.log10(tm)+
cy * Math.log10(td) + cy * Math.log10(tm));
                                currentY*=2;
                                g.setX(d.getX());
                                g.setY(d.getY());
                                g.setGazeX(currentX/10);
                                g.setGazeY(currentY/10);
                                g.setTrackingTime(d.getTrackingTime());
                                lastgaze.setGazeX(currentX/10);
                                lastgaze.setGazeY(currentY/10);
                                lastgaze.setTrackingTime(d.getTrackingTime());
                                System.out.println(g);
                                listgx.add(g);
                        }
                }
                lastgaze.setGazeX(d.getX());
                lastgaze.setGazeY(d.getY());
                lastgaze.setTrackingTime(d.getTrackingTime());
                lastActivityTime = d.getTrackingTime().getMillis();
        }
        return listgx;
}
public class GazeXY {
        long X;
        long Y;
        long gazeX;
        long gazeY;
        DateTime trackingTime;
        public long getX() { return X;}
        public void setX(long x) { X = x;}
```

```
        public long getY() {return Y;}
        public void setY(long y) {Y = y;}
        public long getGazeX() {return gazeX;}
        public void setGazeX(long gazeX) {this.gazeX = gazeX;}
        public long getGazeY() {return gazeY;}
        public void setGazeY(long l) {this.gazeY = l;}
        public DateTime getTrackingTime() {      return trackingTime;}
        public void setTrackingTime(DateTime trackingTime) {this.trackingTime = trackingTime;}
}
```

Appendix V: Sample Javascript Codes for Data Collection

```
var tracker=tracker||{};
tracker.timeout=0;//Micro Seconds
tracker.mousePosition = {};//Mouse Position
tracker.elementPosition = {};//Element Position
tracker.elementMousePosition = {};//Element Position
tracker.scrollKeysArray=[];
tracker.docscrolltop = 0;
tracker.docscrolldelta = 0;
tracker.elemscrolltop = 0;
tracker.elemscrolldelta = 0;
tracker.events=[];
tracker.el = null;//Element To Track Activities
//TIMET
tracker.timer=false;
tracker.stimer=false;
tracker.timerInit=false;
tracker.trackingtime=new Date().getTime();
tracker.regno = "";
tracker.sdiv=null;
tracker.cdiv=null;
tracker.ediv=null;
tracker.lastAjaxObject={};
tracker.init=function(){
  this.timeout=200;
  this.scrollKeysArray=new Array(33,34,35,36,37,38,39,40);
  this.el=jQuery("html");
  //TRACK DOCUMENT SCROLL
  jQuery(window).scroll(function(e){
    clearTimeout(tracker.timer);
    tracker.timer = setTimeout( function () {
      if(tracker.timerInit){
        tracker.captureDocumentScrollStop(e);
      }
    },200);
  });
  tracker.timerInit=true;
  //TRACK MOUSE WHEEL SCROLL
  this.el.on('mousewheel DOMMouseScroll', function (e) {
    if(e.target.tagName.toUpperCase()!=='BODY' &&
e.target.tagName.toUpperCase()!=='HTML'){
      tracker.captureMouseScrolls(e);
    }
    else{
      tracker.captureDocumentScrolls(e);
    }
  });
  var allevents=tracker.getAllEvents(this.el[0]);
  this.el.bind(allevents, function(e) {
    var currentElem = e.target || e.srcElement;
    var currentElemOffset = jQuery(currentElem).position();
    var currentElemPosition = jQuery(currentElem).offset();
    var mousePosition = {};
    switch(e.type){
      case "keydown","keypress","keyup":{
        var key = e.which;// PgUp(33), PgDn(34), End(35), Home(36), Left(37), Up(38),
Right(39), Down(40)
        //Detect Scroll Events
        if(jQuery.inArray(key,tracker.scrollKeysArray) !== -1 ){//}&&
jQuery.inArray(currentElem.tagName,["INPUT","TEXTAREA"])!==-1) {//KEYBOARD SCROLL
          tracker.captureKeyboardScrolls(e);
      }
```

```
            else{//KEYBOARD NAVIGATION
               tracker.captureKeyboardActivities(e,e.type,key);
               //cdiv.html("Key Event : "+new Date().getTime());
            }
            break;
         }
         case "mousemove":{//,"mouseleave","mouseenter"
            //cdiv.html("Mouse Move Event");
            clearTimeout(tracker.timer);
            tracker.timer = setTimeout( function () {
               if(tracker.timerInit){
                  tracker.captureMouseActivities(e,e.eType);
               }
            },200);

            break;
         }
         case "mouseup":{//We are interested in mouse key release which may not signify click or
double click
            tracker.detectTextSelection(e);
            //tracker.cdiv.html("Mouse UP Event : "+new Date().getTime());
            break;
         }
         /*
          case "mouseover":{//,"mouseout"
            if (jQuery(currentElem).hasScrollBar()) {
               jQuery(currentElem).scroll(function() {

                  //if($(this).scrollTop() + $(this).innerHeight() >= $(this)[0].scrollHeight) {
                    //alert("show");
                  //} else {
                     //clearTimeout(tracker.stimer);
                     //stimer = setTimeout( function () {
                        // do stuff
                     //    alert('ELEM STOP Scrolling');
                      //  },150);
                  //}

               });
            }
            tracker.cdiv.html("Mouse Hover Event : "+new Date().getTime());
            break;
         }*/
         case "click":{
            //tracker.cdiv.html("Click Event : "+new Date().getTime());
            tracker.processEvent(e,"click");
            break;
         }
         case "dblclick":{
            if(tracker.detectTextSelection(e)){
               tracker.processEvent(e,"dblclick");
            }
            //tracker.cdiv.html("Double Click Event : "+new Date().getTime());
            break;
         }
         case "contextmenu":{
            tracker.processEvent(e,"contextmenu");
            //tracker.cdiv.html("Context Menu Event : "+new Date().getTime());
            break;
         }
```

```
            case "wheel":{

               clearTimeout(tracker.timer);
               tracker.timer = setTimeout( function () {
                  if(tracker.timerInit){
                     tracker.captureElemScrollStop(e);
                     tracker.processEvent(e,"elem_scroll_stop");
                  }
               },200);
               break;
            }
            case "cut":{
               tracker.processEvent(e,"cut");
               break;
            }
            case "copy":{
               tracker.processEvent(e,"copy");
               break;
            }
            case "paste":{
               tracker.processEvent(e,"paste");
               break;
            }
            case "dragstart","dragenter","drag","dragover","dragleave","dragend":{
               tracker.processEvent(e,"drag");
               break;
            }
            case "select":{
               if(tracker.detectTextSelection(e)){
                  tracker.processEvent(e,"select");
               }
               break;
            }
         }
         if(currentElem.tagName.toLowerCase()!=='body'){
            //tracker.ediv.html("Event : "+e.type+" -> "+currentElem.tagName+" Position :
"+JSON.stringify(currentElemPosition)+" Offset : "+JSON.stringify(currentElemOffset)+" Mouse :
"+JSON.stringify(mousePosition));
         }
      });
   };
   tracker.captureDocumentScrolls=function(e){
      //tracker.cdiv.html("Document Scroll ");
      tracker.processEvent(e,"document_scroll");
      //sourceElemPosition alert('Stopped Scrolling');
   };
   tracker.captureDocumentScrollStop=function(e){
      //http://jsfiddle.net/pH93b/18/
      e.target.tagName=(e.target.tagName===undefined)?"html":e.target.tagName;
      var currentscrolltop=jQuery(e.target).scrollTop();
      tracker.docscrolldelta=currentscrolltop-tracker.docscrolltop;
      tracker.docscrolltop=currentscrolltop;
      tracker.processEvent(e,"document_scroll_stop");
      //sourceElemPosition alert('Stopped Scrolling');
   };
   tracker.captureElemScrollStop=function(e){
      e.target.tagName=(e.target.tagName===undefined)?"html":e.target.tagName;
      var currentscrolltop=jQuery(e.target).scrollTop();
      tracker.elemscrolldelta=currentscrolltop-tracker.elemscrolltop;
      tracker.elemscrolltop=currentscrolltop;
      tracker.processEvent(e,"document_scroll_stop");
   };
```

```
  tracker.captureMouseScrolls=function(e){
    tracker.processEvent(e,"mouse_scroll");
  };
  tracker.captureKeyboardScrolls=function(e,eType,hotKey){
     tracker.processEvent(e,"keyboard_scroll");
  };

  tracker.captureKeyboardActivities=function(e,eType,hotKey){
     tracker.processEvent(e,"keboard_activity");
  };
  tracker.captureMouseActivities=function(e,eType,hotKey){
     tracker.mouseData(e);
     tracker.processEvent(e,"mouse_activity");
  };

  tracker.mouseData=function(e){
     if(e!==null && e!==undefined){
     if (e.pageX || e.pageY) {
        tracker.mousePosition.X = e.pageX;
        tracker.mousePosition.Y = e.pageY;
     } else {
        tracker.mousePosition.X    =    e.clientX    +    (document.documentElement.scrollLeft    ||
document.body.scrollLeft) -
           (document.documentElement.clientLeft || 0);
        tracker.mousePosition.Y    =    e.clientY    +    (document.documentElement.scrollTop    ||
document.body.scrollTop) -
           (document.documentElement.clientTop || 0);
     }
     }
  };
  tracker.getElementXY=function(e){
     console.log(e.type);
     if(e.type!=="scroll" && e.type!=="DOMMouseScroll"){
        var element=jQuery(e.target);//.offset();
        var rect = element.get(0).getBoundingClientRect();
        var scrollTop = document.documentElement.scrollTop?
                  document.documentElement.scrollTop:document.body.scrollTop;
        var scrollLeft = document.documentElement.scrollLeft?
                  document.documentElement.scrollLeft:document.body.scrollLeft;
        var elementLeft = rect.left+scrollLeft;
        var elementTop = rect.top+scrollTop;

        if (document.all){ //detects using IE
           x = event.clientX+scrollLeft-elementLeft; //event not evt because of IE
           y = event.clientY+scrollTop-elementTop;
        }
        else{
           x = e.pageX-elementLeft;
           y = e.pageY-elementTop;
        }
        tracker.elementPosition.L=elementLeft;
        tracker.elementPosition.T=elementTop;
        if(!isNaN(x)){//MOUSE HAS NOT MOVED!
           tracker.elementMousePosition.X=x;
           tracker.elementMousePosition.Y=y;
        }
        else if(tracker.elementMousePosition.X==undefined){
           tracker.elementMousePosition.X=-1;
           tracker.elementMousePosition.Y=-1;
        }
     }
  };
```

```
tracker.processEvent=function(e,type){
   tracker.getElementXY(e);
   var ajaxData={};
   ajaxData.element=e.target.tagName;
   ajaxData.event=type;
   ajaxData.mousePosition=JSON.stringify(tracker.mousePosition);
   ajaxData.elementPosition=JSON.stringify(tracker.elementPosition);
   ajaxData.elementMousePosition=JSON.stringify(tracker.elementMousePosition);
   ajaxData.docscrolltop=tracker.docscrolltop;
   ajaxData.docscrolldelta=tracker.docscrolldelta;

   ajaxData.elemscrolltop=tracker.elemscrolltop;
   ajaxData.elemscrolldelta=tracker.elemscrolldelta;

   tracker.trackingtime=new Date().getTime();
   if(JSON.stringify(tracker.lastAjaxObject)!==JSON.stringify(ajaxData)){
      tracker.processAjax(ajaxData);
   }
};

tracker.processAjax=function(ajaxData){
        ajaxData.trackingtime=tracker.trackingtime;
        ajaxData.sessionkey=tracker.sessionkey;
        var formURL = "addwine";
        $.ajax(
        {
           url : "http://localhost/tracker/track",
           type: "POST",
           async: true,
           crossDomain:true,
           data : ajaxData,
           success:function(data, textStatus, jqXHR)
           {
              jQuery("#rdiv").html(data);
              //data: return data from server
           },
           error: function(jqXHR, textStatus, errorThrown)
           {
              //jQuery("#rdiv").html(jqXHR.responseText+":"+textStatus+" : "+errorThrown);
              //if fails
           }
        });

};
```

Appendix VI: Sample Question and Answer Questions used in Data Collection

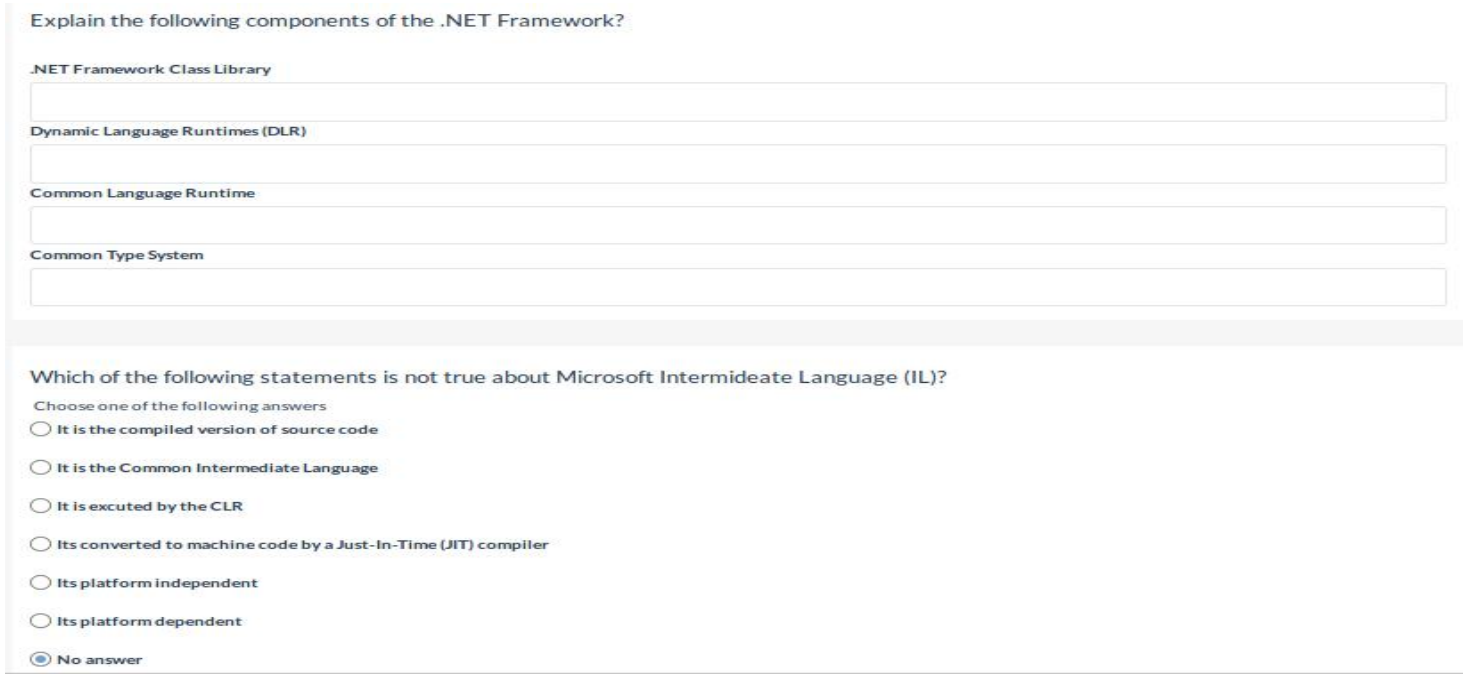

Figure A1 Sample Survey Questions

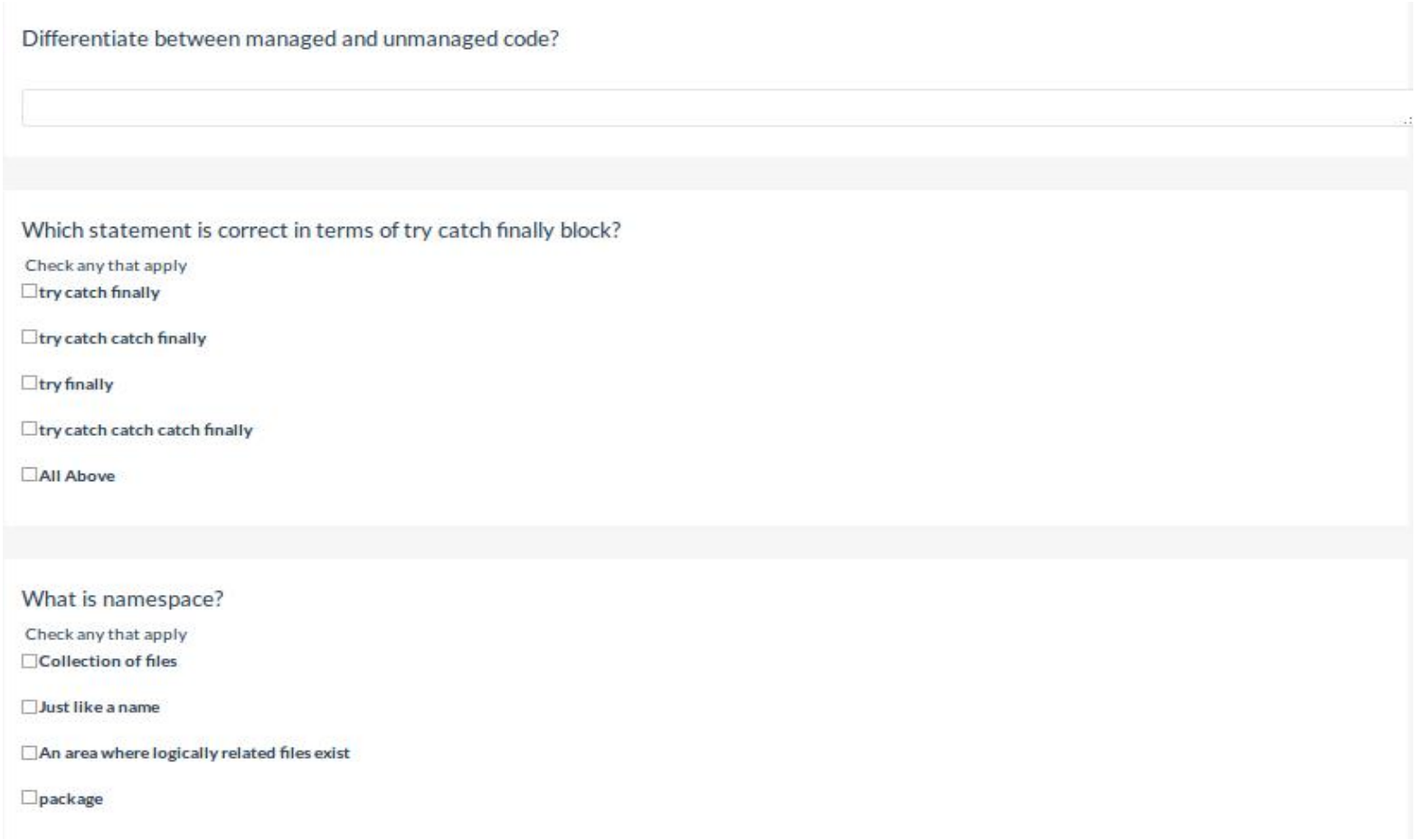

Figure A2 Sample Survey Questions

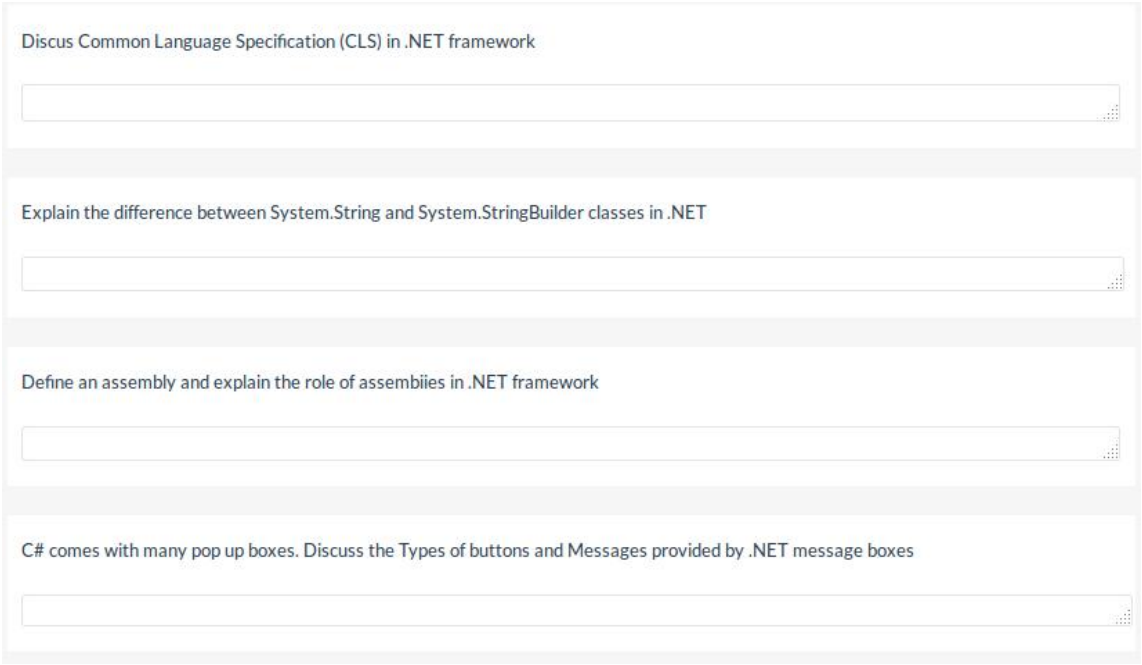

Figure A3 Sample Survey Questions