Department of Computer Science and Engineering (CSE)
Islamic University of Technology (IUT)
Organization of Islamic Cooperation (OIC)

M.Sc. Thesis

Web Interaction Model with Multimodal Interface using Speech and Head Gestures

By

Mountapmbeme Aboubakar

Student ID: 154604

Supervised By:

Dr. Md. Kamrul Hasan
Associate Professor,
Department of Computer Science and Engineering, IUT

May, 2018

RECOMMENDATION OF THE BOARD OF EXAMINERS

The work titled "**Web Interaction Model with Multimodal Interface using Speech and Head Gestures**", submitted by **Mountapmbeme Aboubakar (Student Id. 154604)** in the academic year 2017-2018 has been found satisfactory and accepted as partial fulfilment of the requirement for the degree of Master of Science in Computer Science and Engineering (M.Sc. Engg.(CSE)) on Thursday, 31 May 2018

DECLARATION

This work, Web Interaction Model with Multimodal Interface using Speech and Head Gestures, is the output of the research carried out by Mountapmbeme Aboubakar (Student Id. 154604) under the supervision of Dr. Md. Kamrul Hasan (Associate Professor, CSE Department, IUT) and the co-supervision of Mr. Hasan Mahmud (Assistant Professor, CSE Department, IUT). Completed and submitted to the Department of Computer Science and Engineering (CSE), in Partial Fulfilment of the Requirements leading to the award of the degree of Master of Science in Computer Science and Engineering (M.Sc. CSE).

Mountapmbeme Aboubakar

Student Id: 154604

Mr. Hasan Mahmud

Assistant Professor, CSE, IUT

Dr. Kamrul Hasan

Associate Professor, CSE, IUT

DEDICATION

To my late father, DONMY SEIDOU and my mother, MEFIRE PASMA.

ACKNOWLEDGEMENT

All the praises and thanks be to Allah, the Lord of the Alamin (mankind, jinn and all that exists). Qur'an (V. 1:2).

First and foremost I thank the Almighty Allah (SWT) for blessing us with this beautiful thesis work and for giving us the knowledge, strength, determination, spirit of team work and for accompanying us throughout the successful completion of this thesis.

My special thanks goes to Dr. Md Kamrul Hasan and Mr. Hasan Mahmud, my thesis supervisor and co-supervisor, who directed and supported me throughout the successful completion of this thesis.

I express my sincere gratitude and thanks to Prof. Dr. Muhammad Mahbub Alam, head of the Computer Science and Engineering Department of the Islamic University of Technology and the entire staff of the department for the work and efforts they are doing to impart knowledge to the students in general.

Abstract

Today, almost every individual in the society requires the internet to complete one task or the other, be it in education, professional life or social life. These individuals with diverse abilities and limitations cannot by pass the internet in certain cases. However, there are a group of people with special needs who have difficulties in efficiently accessing information on the web due to some limitations in using the standard methods of interacting with computers. Because of their limitations, alternative interaction techniques are usually designed purposely to address these shortcomings. In an effort to make access to the World Wide Web all-inclusive irrespective of people's abilities, we have developed a multimodal input interfaces that combines head movements and voice commands. This interface acts as an alternative to the standard mouse especially for people with upper limb motor disabilities. We designed, developed and evaluated our multimodal interface. We built predictive models for our system to act as bases for designers. We constructed a Fitts' law model to predict the pointing time of our system. We equally built a KLM model for the whole system to be used to predict the time needed to complete a task using this interface. With our initial interest being giving easy access to information on the web to people with disabilities, we tested our interface on two of the most common web browsing activities today, namely e-commerce and social networking. We compared the performance of our system in these web navigation scenarios to the performance of the standard mouse and the voice-only navigation. Our system performed better than the voice-only navigation and it equally addressed some of the limitations of voice-only interaction such as ambiguity and long list of unstructured command vocabulary.

Keywords

Human Computer Interaction (HCI), Interaction techniques, Multimodal Interaction, Speech Interfaces, Head Tracking.

Table of Contents

List of Tables

List of Figures

1. Introduction

Nowadays, many people turn to the internet to obtain information that is useful to them in one way or the other. These people have diverse capacities and abilities to use the web effectively. Among these people, there are special groups such as people with motor disability, visual disability, hearing disability and other disabilities who cannot use the traditional and standard methods of interaction with computers. Because of this, they often face difficulties such as navigating between and within web pages using the mouse or have incapacities doing text entry on forms with the standard keyboard. In the past decades, there have been research work to make the web all-inclusive for everyone in the society [1]. Most of these research works [2] [3] mainly focused on developing alternative interaction methods to the standard keyboard and mouse. As a result, surrogate unimodal interaction techniques were developed for the people, especially for the people with special needs, to interact with computers. Most, if not all of these alternative techniques of interacting with computers make use of the opportunities provided by the different modalities or natural ways with which humans communicate with each other. Some of these methods are gesture-based interactions, speech-based interactions, brain-computer interface (BCI), electromyography-based interactions and many others. These were developed as substitutes to the traditional input devices like the keyboard and mouse.

As already mentioned, speech is one of the alternative methods that has been used for interacting with computers, especially by people with upper limb disabilities. Speech/voice commands have been used as a unimodal input method for web navigation [1] [2]. A lot of progress has been done and interaction using speech has greatly improved. However, speech interaction still presents some challenges as compared to the standard mouse and keyboard [2] [3].

A recent trend in novel interaction methods is the use of multimodalities. In multimodal interaction techniques, one or more ways with which humans communicate are combined into one interaction model. Recently, the CHI SIG [3] has investigated and recommended the use of speech in combination with other interaction modalities to create natural, intuitive and easy to use multimodal interfaces. It is believed that, using the different interaction techniques in combination will complement one another and help increase accuracy, effectiveness and naturalness leading to an overall increase in usability and user experience. This recommendation, presented in the CHI conference proceedings of 2017 is based on the observation that humans communicate effectively in the real world by using the combination of their five senses. As such, there is a call to research on possible multimodal interaction techniques involving speech that will address some of the challenges of speech interfaces such as slow information processing, ambiguity in spoken commands, unstructured and long list of command vocabulary, visual feedback, pointing and selection of objects on the screen, and the cognitive overload imposed on the user when using speech interfaces [1] [2] [3] [4].

1.1 Objective

Based on the above discussion, our objective is to design, develop, and evaluate a multimodal interface combining voice commands and head gestures for navigating the web that will address some of the challenges of speech-only interfaces and at the same time be natural and intuitive to use. This model is targeted at people with permanent or temporary upper limb motor disabilities who cannot use the standard mouse and keyboard. In this work, we do the following:-

a. Design and develop a multimodal input interface combining voice commands and head gestures from an Interaction Design perspective for easy navigation and improved user experience. We try to build a pointing interface similar in function to the mouse that uses head movements and voice commands as a means of interaction.
b. Investigate the use of speech and head gestures as a multimodal interaction technique for navigating web pages and compare this with unimodal voice-based navigation and mouse-based navigation. Our target is to make accessing information on the web through web browsers easy for people who cannot use the mouse. In order to test the usability of our system on web browsing and navigation, we conduct two experiments on common web browsing habits that involves moving through and within web pages on a standard web browser such as Mozilla Firefox.
c. Finally, evaluate the overall model following HCI methodologies. Here, we construct HCI models to validate our multimodal interface. These models also serves as bases for comparison with similar systems.

1.2 Motivation

Our prime motivation is to make the web easily accessible to everyone irrespective of their abilities and limitations. People mostly access information on the web through a web browser by using a mouse and/or keyboard. However, this is not the case for those with disabilities who cannot manipulate a mouse or a keyboard. Therefore, in order to provide them with equal opportunities, we try to build an alternative interaction technique to the standard mouse that is adapted to their abilities and limitations. We were motivated by the use of speech in web browsing [2] and in multimodal interfaces [3]. We are interested in speech based interaction and its role in multimodal interface design and its application in web browsing and navigation. In [2], an attempt was made to use speech as a unimodal input for web navigation. However, it presented some challenges such as ambiguity in voice commands and long list of unstructured commands. Some voice browsers such as Conversa were developed for speech navigation. Today however, we cannot find voice browsers that take speech input and render visual output. Even Conversa is not available today. After going through the literature and realising the challenges facing unimodal speech interfaces and the absence of a suitable voice browser, we felt a concern for the people with special needs. By exploring the opportunities speech presents in multimodal interfaces as described in [3], we sought to harness these opportunities in a multimodal interface to address some of the challenges of speech-only interaction and to provide an alternative interaction method for the people with upper limb motor disabilities.

1.3 Contribution

Our primary contribution involves the design, implementation and testing of a speech-based multimodal input interface that makes use of head movements to address the challenges of unimodal speech input method. In our interface, we introduce a virtual mousepad that works similar to the physical mousepad. Our virtual mousepad allows for customisation in relation to the user's screen and the position of the head. The virtual mousepad also incorporates a mechanism to counter the inertia associated with head movements and to mimic the way of operation of a physical mousepad or real mouse. We equally provide a visual feedback screen where the user can monitor and adjust or reset his/her movements to his/her convenience. The articulations in the interaction model of our multimodal input interface consists of head movements and voice commands. This helps address the problems of articulations in unimodal speech model where the speech input is sometimes unstructured [2]. Though we did not actually test out interface with amputees or users with special needs, our system is specially designed considering their abilities and limitations. We had normal users test the system in two experiments. The users interacted with the system without any external interference. Users with special needs might have some cognitive development that may give them a special advantage in using the system. We have plans in our future work to conduct experiments that involves this group of users. Nevertheless, the participants in our experiments gave positive feedback about the usefulness and comfort of our interface.

We are also interested in exploring the huge possibilities that multimodalities provide in the design and implementation of novel interaction methods.

2. Background and Related Works

2.1 Speech/Voice Interaction

Using speech or voice for interacting with computing systems has been around for decades and is a subject of continuous research work due to the opportunities speech presents as a natural and intuitive way for people to communicate with computers. Speech user interfaces use spoken language/commands for user input and/or output [5]. In speech interfaces, when speech is used as input, it is referred to as speech recognition and when it is used as output it is referred to as speech synthesis (text-to-speech). A speech interface can be made of speech recognition or speech synthesis or both. Spoken speech is recorded through the microphone. Speech recognition algorithms are used to decode the inputted command into text which is further processed and used to determine users' interaction with the system and generate a response. In systems that use speech synthesis, text is converted into speech by speech synthesis engines and passed to the user through normal audio output of the system [5].

In [5], characteristics of speech interfaces and the implications of testing speech interfaces are discussed. As mentioned in [5], speech technology is not a new concept. However, the use of speech as an interface presented so many challenges and was often blamed for the poor user experience. One of the greatest challenges which somehow still exists today is accurately decoding the spoken input from the user. This is because human language comes in different varieties and forms. This is further complicated by the fact that the way people speak (accents) varies from one individual to another, making it difficult to build universal speech recognition systems. This various in accents and low recognition accuracy in speech interface is amplified by the use of long and unstructured commands in speech interaction. We designed our system to support a short and concise list of recognisable voice commands. The voice commands selected in our system are English words and phrases common in everyday computer jargon. These words are easy to pronounce, to learn and to remember and we are acquainted with the basic operations of a mouse. By combining these short command list in a multimodal setup with head movements, achieve a multimodal pointing interface that can perform almost every action an ordinary mouse can do. Thus giving a way for speech to be used in a more structured manner in interactions. To go further, we have plans in our future works to make the list of voice commands customization by the user. This will introduce more flexibility and reduce the pain of having to cope or adapt with words or phrases that we found difficult.

Today, the web is an important source of information in the society. Information on the web is accessed via a web browser which visually presents its contents to the users. Traditionally, users interact with the browser by the use of a mouse and/or keyboard. One of the first domain of using speech as an interaction method was in web navigation [1] [4] [6] . Speech recognition and synthesis have been used as alternative methods for accessing information on the World Wide Web particularly for people who cannot use the standard mouse and keyboard. On one hand, this lead to the birth of voice browsers. As defined in [2], a voice browser is a web browser with a least one of the following capabilities:

1. Can render web pages in an audio format (speech generation)
2. Can interpret spoken input for navigation (speech recognition)

In a web browser that accepts voice input and renders visual output, a user navigates by speaking the text of a link or an associated number instead of clicking with the mouse. Conversa by Conversational Computing was a voice browser that provided this feature. One of our main objective is to leverage a speech-based interface for navigating and browsing information on visual web browsers. We are motivated by the fact that after our search, we could not lay hands on a speech-enabled visual web browsers. Conversa was one such web browser. However, we do not even find Conversa today. Thus, given a multimodal speech-based interface an ordinary web browser can be used to navigate web pages use voice commands. This will be helpful to users with special needs who cannot manipulate the mouse or keyboard.

In [2], a comparative study was carried out to compare the effectiveness of voice controlled vs mouse controlled web browsing. This experimented on the use of speech as a unimodal input to web browsing. This study found out that the mouse was faster as expected with the voice controlled browsing adding about 50% to the task completion time. It was realised that users had difficulties selecting items from a list of hyperlinks when using voice commands. The long nature of the text of these hyperlinks also complicated the situation. Numbered links were used to try to improve the performance of voice navigation [2]. With numbered links, a user is required to utter a number attached to a link in order to activate the link. However, the experiment conducted by [2] exposed that voice navigation by reading text of links is better than using numbered links. Using numbered links add an extra cognitive step to the users who have to determine the numeric value associated with the desired link before they can speak the number and activate the link. To solve the problem of having to read out long text of links or remove the cognitive burden of using numbered links, our multimodal input system by combining head movements and voice commands operates in a similar manner as the physical mouse. As a point-and-click interface, selecting from a list of hyperlinks will be easier with voice commands as the user will only have to point to the item of interest using head movements and then activate the link by simply uttering a short voice command such as “left click”.

Though in [2] it is argued that the characteristics of web navigation makes it suitable for voice commands because it might have a relatively simple and small set of command vocabulary such as “go back”, “follow link”, “refresh”, etc., this might not be the case in practice. In an assessment of voice browsing for older adults, especially those new to computers, it was found that users use a faulty mental model to direct computers which leads to a mismatch between expectations and task demands [4]. Hanson et al [4] conducted an experiment where they asked participants to control a web browser by voice. They found that it was rather difficult for participants to say the short text presented on labels and buttons. Rather, participants tended to describe what they wanted to do or simply pointed to the screen to indicate their preferences. Participants simply wanted to say what they wanted to do instead of reading out the text on a button or link.

Hanson et al [4] also conducted an experiment to explore the possible use of Internet Explorer and Firefox by voice control. This time, they provided a printed list of commands to be used by participants and participants were free to suggest better commands. Though this approach was better than the open ended approach discussed before, participants constantly referred to the list. Participants had difficulties selecting links or targets within a web page. They were found to constantly use mouse analogies to execute tasks. As described in [4], some tended to say “click go” to select a Go button or “click weather” to select a weather link. It is reported that one participant even went as far as using a command phrases “mouse left”, “mouse down” to move the mouse around the page to targets. Another problem highlighted in [4] is that of ambiguity. As reported, the command “Home” turned to be a confusing term because sometimes participants referred to the website’s home page and sometimes they referred to the browsers Home button. What we observe here is that participants have a correct mental model based on the mouse for navigating and selecting targets on the screen. However, they lack an equivalent of a mouse pointer to assist them in easily completing tasks on visual interfaces. For example, having a visual feedback indicating which of the “Homes” the user wishes to select just prior to echoing the “Home” command will solve the problem of ambiguity and probably speed up the task. In order words, having an equivalent of highlighting a target or object upon mouse pointer hover on top of the object before performing a desired action on the object. After reading about this attitude of users always trying to use mouse analogies when directing a computer by voice, we came up with the idea of designing a voice-based multimodal pointing input interface that mimics the mouse. Since the user is focused on the screen with the direction of the head and gaze pointing to the target of interest, we exploit this natural setup or feature and use as a means to control the mouse pointer on the screen. Head or eye tracking have been used to control the mouse pointer. With the user able to control the mouse pointer with his/her head and point to an object to interest, he can then give a voice command directly to the computer which will then execute the desired action on his/her focused object of interest. Because the design of this interface draws on the mouse analogy, we believe it will be easy to learn and use by people, especially people who have some acquaintance in the functioning and operation of the mouse. Our speech-based multimodal interface will also address the problem of ambiguity explained in [4]. Because the user is not required to read the text on an object in order execute an action on that object, the problem of ambiguity is resolve. With our system, if more than on object on the screen have the same foreground text, the user will point to the object of interest using head movements and perform the desired action with the available voice command.

Gruenstein et al [1] presents a toolkit for developing, deploying and evaluating web accessible multimodal user interfaces where users interact using speech, mouse, pen and/or touch. They argue that it is often difficult to make multimodal systems incorporating speech available to the large public because the systems usually rely on a combination of components such as speech recognition engines, natural language processing components and other components making it difficult to distribute to users, especially users with limited computer knowledge. They present a client-server architecture for speech recognition and synthesis for developing web-based multimodal applications that will allow developers to cheaply develop, deploy and evaluate multimodal interfaces. However, their focus is on technical and infrastructural concerns rather than design concerns or navigation problems faced when using speech command to browse the web. Our work focuses on design constrains, simplicity of voice-commands in interaction as well as user concerns, abilities and limitations in using speech-enabled interfaces to navigate the web. In order words, we focus more on the user and his/her abilities. Gruenstein et al [1] have developed a visual game consisting of grids and images. The user moves the images to appropriate slots on the grid by uttering command sentences such as “Take the sheep and drop it into the fifth square”. This speech interaction structure is designed specifically for this game and does not provide a general method for interacting with visual interfaces. Outside of this game, such commands will not work except if a new application using the WAMI toolkit is developed for that specific task. When we compare with the standard mouse, we immediately notice that the mouse is a general purpose input method that works on almost, if not all sorts of visual interfaces. The toolkit provided in [1] allows developers to develop specific applications that make use of speech as a modality of interaction. Thus with the WAMI toolkit, we cannot have a general purpose speech interaction technique that will allow users to interact with all kinds of visual interfaces. Our speech-based multimodal input interface is designed to be a general purpose interaction method that can be used in any kind of visual interface that supports the functions and operations of a standard mouse.

Dubinsky et al [7] presents an approach for developing and evaluating speech interfaces centred on the user. It states that the evaluation process exists as long as the product development process is and the users together with experts in human-computer interaction are involved with the evaluation and improved design. It starts by reviewing the existing user interface evaluation techniques. Two of these techniques, the cognitive walkthrough and the heuristic analysis techniques are non-user evaluation methods. That is they take place in the absence of the user. As explained in [7], the cognitive walkthrough technique is mostly used to explore what the user thinks when first using the interface without any training. Meanwhile in the heuristic analysis technique, the user interface is evaluated according to ten defined heuristics that guide us with the design decisions. The other two techniques require the participation of users. In the Wizard of Oz technique, we fake as if the system exists and simulate an experiment with real users. Somebody behind the scenes or sometimes visible to the users perform the actions as directed by the users and provide feedback where possible. The main advantage of this technique is that there is no need for the real implementation of the system. The last method involving user participation is the Thinking Aloud method. In this technique, we ask the user to speak out his/her own thoughts while using the interface as we observe. We then analyse the user's actions and thoughts and check whether the sequences fit and expose interface problems that cannot be revealed by the examination of the user actions alone. Dubinsky et al then describe their method of user-based development of speech interfaces. This method is based on the well-known agile approach of software development. They suggest an iterative and inductive method with three main principles. The first principle is user collaboration. It prescribes continuous user collaboration to obtain feedback from the system. The second principle involves dealing with changes in the users' needs as feedback is continuously received. The last principle they defined in their proposed method requires users to assist in defining usability automated tests that enable fast feedback to developers.

Though it is generally agreed that users with temporary or permanent motor impairments will benefit from using speech as an input method, it however presents some challenges. Using speech as input and/or output affects users' performance in a task because formulating speech input and/or processing speech output consumes the user's short-term and working memory which conflicts with tasks such as planning and problem solving [2] [4] [8] . We believe by providing a short list of simple and concise voice-commands in a multimodal setup will alleviate most of the burden of formulating long sentence on the part of the user.

Another challenge is the recognition error which in most cases is inevitable especially when the command phrases are long and unstructured. The more unstructured and lengthy a voice command is, the more it is prone to recognition error and this tend to have a great effect on task completion time [9]. The best tasks for speech input are those in which the user has to issue brief commands using a small vocabulary. Our system follows this pattern. We have a short list of command vocabulary which is easy and simple to learn and to remember.

As explained in [10] lot of progress has been made in the speech recognition domain and recognition accuracy is improving day-by-day. This has been made possible by research in machine learning [3], particularly the use of neural networks [10] as well as availability of computing resources and huge amount of training data [5]. This has brought speech recognition to near human levels of performance [10]. However, as explained in [3], achieving 100% speech processing may not even be required. It argues that proper interaction design can complement speech processing in ways that compensate for its lack of accuracy. This includes considering using speech in multimodal settings to improve its accuracy and performance as well as address challenges of voice-only interfaces.

2.2 Multimodal Interface and Speech/Voice interaction

Multimodal interaction techniques are those in which one or more communication modalities (e.g speech, touch, gesture, etc) are combined into one interaction model. Recently, the CHI SIG [3] has investigated and recommended the use of speech in combination with other interaction modalities to create natural, intuitive and easy to use multimodal interfaces. It is believed that, using the different interaction techniques in combination will complement one another and help increase accuracy, effectiveness and naturalness leading to an overall increase in usability and user experience

It is assumed that multimodalities with speech as well as good design will help address some of the challenges of speech interaction and improve accuracy as well as increase naturalness in the interface [3] [11]. The CHI 2011 conference had a workshop on the future of natural user interfaces [11]. The aim of the workshop was to propose an integrated approach for designing and developing multimodal Natural User Interfaces (NUI) that involves speech, touch, gestures etc. There is ongoing concern to leverage natural methods of human communication in combination with one another to create easy to use, intuitive and natural interaction techniques [3] [11]. It is argued that while a lot of outstanding work has been done on each of the interaction modalities, little attention has been given to work on combining these modalities together. In [11] Natural User Interfaces are defined as interfaces that enable users to interact with computers in the way we interact with the real world. As defined in [11] a mode of interaction is a given combination of a type of input (e.g. touch) and output (e.g. visual). A multimodal interaction refers to either a combination of more than one output or more than one input as part of the interaction technique. For the interaction experience to be more natural, we have to mimic the ways humans communicate in the real world. In the natural world, humans interact using a combination of multiple modalities. Users with special needs will get the most benefits from these alternative interaction techniques. The fact that access to digital content needs to be provided to everyone irrespective of their ability and/or limitation calls for a reconsideration of the “one size fits all” paradigm which in most cases only serves normal or ordinary users who do not have problems using the traditional and well known methods of interaction.

To summarise, speech as a modality will be beneficial to users with special needs especially for users with upper limb motor impairments who cannot use the standard mouse and/or keyboard on visual interfaces. However, speech input alone presents some challenges that can be attenuated by designing multimodal interfaces that combine speech with other modes of interactions. Multimodality offers the opportunity for developing natural, intuitive and easy to use interfaces.

As a result of the above findings, in our effort to provide access to the web and other visual interfaces in general to users with upper limb disabilities, we set forth to design a multimodal input interface that combines voices commands and head movements to create an alternative to the standard mouse for use as input method on visual interfaces.

2.3 Head tracking to control the mouse pointer

As mentioned earlier, gesture-based interaction methods have been used as alternative interaction techniques. In particular, head movements have been tracked and used as input to computer interfaces [12] [13] [14] [15].

Al-rahayfeh et al [16] provides a concise and rich state-of-the-art survey on eye tracking and head movement detection with application areas. As described in [16], there are basically two types of methods through which head movement detection is carried out. We have camera-based or computer vision-based head movement detection and sensor-based head movement detection techniques. Camera-based techniques make use of an RGB camera to capture video frames of the user. Then image processing and computer vision algorithms are applied on the video frames to track the user's head. On the other hand, sensor-based methods make use of sensors attached to the user's head to track the motion and location of the head. The sensors commonly employed are accelerometers and gyroscopes. Other head tracking methods are based on acoustic signals. They detect the user's voice to estimate the direction of the head. Other techniques are hybrid methods that combine these approaches into one system. Though camera-based approaches are usually cheaper and easier to deploy, they however, are less accurate than their sensor-based counterparts.

In [17], Anjankar et al use an RGB camera to track head movements to control the mouse pointer. The paper focuses on algorithms for face detection. Anjankar et al use two algorithms to detect a user's face. One algorithms involves calculating the geometric features of the nose, mouth and chin to detect the face and the other algorithm uses Haar Classifier algorithms for face detection. The authors after detecting a face, use a mean shift tracking technique to move the mouse pointer around the screen. The authors do not elaborate on the implementation of the tracking technique.

In our multimodal system, we use an RGB camera to track the users head movements to control the mouse pointer. We introduce a virtual mousepad that functions similar to the physical mousepad. Our design also allows for the control of the inertia associated with head movements by introducing a break and reset movement mechanism where the position of the mouse pointer on the screen is not directly tied to the position of the user’s head. It allows the user to pause in a course of motion and reset his/her head without affecting the position or distance already covered by the mouse pointer. The implementation of our system focuses on the characteristics of user head movements to produce an interface that is ergonomically convenient for the users.

Eye tracking has been used as an alternative method to head movement tracking for the control the mouse pointer [18]. However, when compared to head movement tracking, the accuracy of eye tracking is still weak and there is ongoing research on robust eye tracking mechanisms [16].

In the upcoming sections, we will introduce and describe our proposed input interface that combines voice commands and head movements to create a multimodal input system as an alternative input method to the standard mouse on visual interface for use by people with upper limb motor disabilities. We are particularly interested on the performance of our system on visual web navigation and browsing.

Before introducing our system, we will describe in the next section some of the HCI methodologies and design models used in our research and experiments.

3. Prerequisites

In this section, we summarise the theoretical knowledge needed to understand the remaining sections of this reports. These are HCI research methodologies and details can be found in [19]. We discuss the Keystroke-Level Model (KLM), the Fitts' law and Analysis of Variance (ANOVA). If you are familiar with this concepts, you can skip to the next chapter.

3.1. Keystroke-Level Model (KLM) and Fitts' law

Designing interaction techniques in HCI usually involves modelling. Models allow us to explore a problem in depth and to understand the problem space easily. Predictive models are widely used to model interaction techniques. As defined in the book Human-Computer Interaction an Empirical Research Perspective by I. Scott Mackenzie [20], a predictive model is an equation that predicts the outcome of a variable based on the value of one or more other variables (predictors). The outcome variable is the dependent variable, typically the time or speed of accomplishing a task. In building models, we aim to build models of interaction that (a) embed a theoretical account of the underlying human processes and (b) serve as prediction tools for follow-up analyses of design alternatives. Examples of predictive models include the Keystroke-level model (KLM) and the Fitts' law.

3.2. Linear regression model

Modelling by building the predictive equation using linear regression is described in [20] as follows.

The basic prediction equation expresses a linear relationship between an independent variable (x, a predictor variable) and a dependent variable (y, a criterion variable or human response)

$$y = mx + b \qquad (1)$$

Where m is the slope of the relationship and b is the y intercept.

Figure 1 below shows the relationship between an independent variable, x and a dependent variable, y.

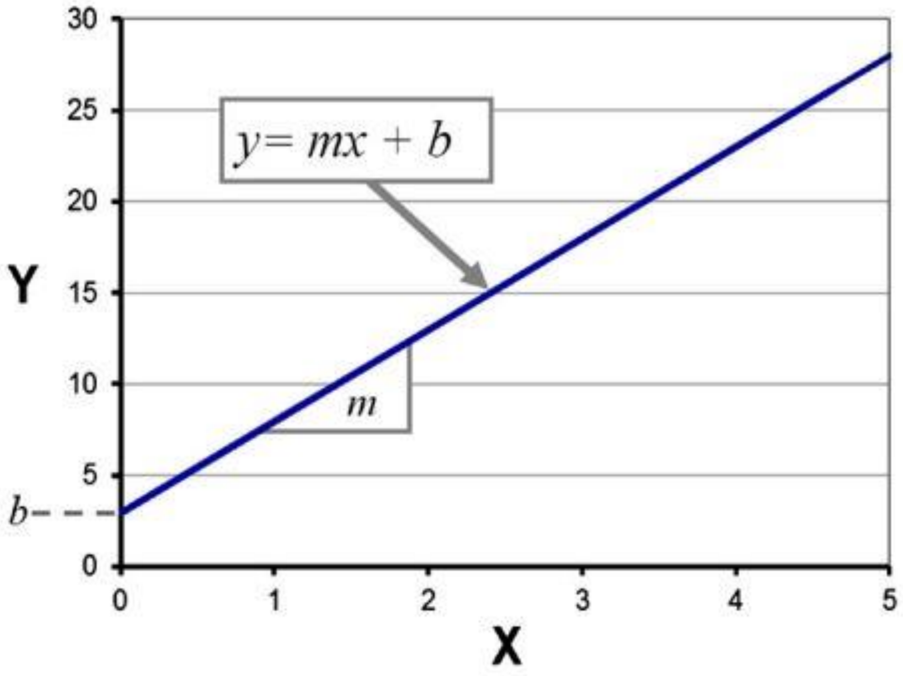


Figure 1: Linear relationship between an independent variable (x) and a dependent variable (y) (Note: Figure taken from I. Scott Mackenzie)

To build this equation, we first need a set of x-y sample points. Although any two ratio-scale variables will do, most commonly the x-y points combine the setting on an independent ratio-scale variable (x) with the measured value of a human response on a dependent variable (y). Since we are dealing with humans, variability is unavoidable, so the sample points are unlikely to lie on the line. They will scatter about. If the model is good, the points will be reasonably close to a straight line. How close is a key question.

Finding the best fitting straight line involves a process known in statistics as linear regression. The objective is to find coefficients m and b in Equation 1 for the line that minimizes the squared distances (least squares) of the points from the line. The result is the prediction equation—an equation that gives the best estimate of y in terms of x. Of course, the model is based on the sample points used in building the equation. If the data are good and the model is correct, a good prediction equation should emerge. There is also a built-in assumption that the relationship is linear, which is not necessarily the case.

Linear regression is used in our Fitts law experiment to build a prediction equation for the task completion time of our multimodal input system, given a target size and distance to reach the target.

3.3. Fitts' Law

We used Fitts' law to model the pointing time of our multimodal input system by building a prediction equation. A brief explanation of Fitts' law as summarised from [20] follows.

One of the most widely used models in HCI is Fitts' law. If an interaction involves a rapid-aimed movement, such as moving a finger or cursor to a target and selecting the target, there's a good chance someone has experimented with the interaction and used Fitts' law to model it. Fitts' law has three uses in HCI: (1) to learn if a device or interaction technique conforms to the model by building the prediction equation and examining the correlation for "goodness of fit," (2) to use the prediction equation in analyzing design alternatives, or (3) to use Fitts' index of performance (now throughput) as a dependent variable in a comparative evaluation.

Fitts argued that the amplitude of an aimed movement was analogous to the information in an electronic signal and that the spatial accuracy of the move was analogous to electronic noise. Furthermore, he proposed that the human motor system is like a communications channel, where movements are like the transmission of signals. Fitts' analogy is based on Shannon's Theorem 17, expressing the information capacity C (in bits/s) of a communications channel of bandwidth B(in s−1or Hz) as

$$C = B\ \log_2\left(\frac{S}{N} + 1\right) \qquad (2)$$

where S is the signal power and N is the noise power.

Fitts presented his analogy—now his "law"—in two highly celebrated papers, one in 1954 (Fitts, 1954) and the second in 1964 (Fitts and Peterson, 1964). The 1954 paper described a serial, or reciprocal, target acquisition task where participants alternately tap on targets of width W separated by amplitude A (see Figure 2 below). The 1964 paper described a similar experiment using a discrete task, where subjects selected one of two targets in response to a stimulus light. It is easy to imagine how to update Fitts' apparatus with computer input devices and targets rendered on a computer display.

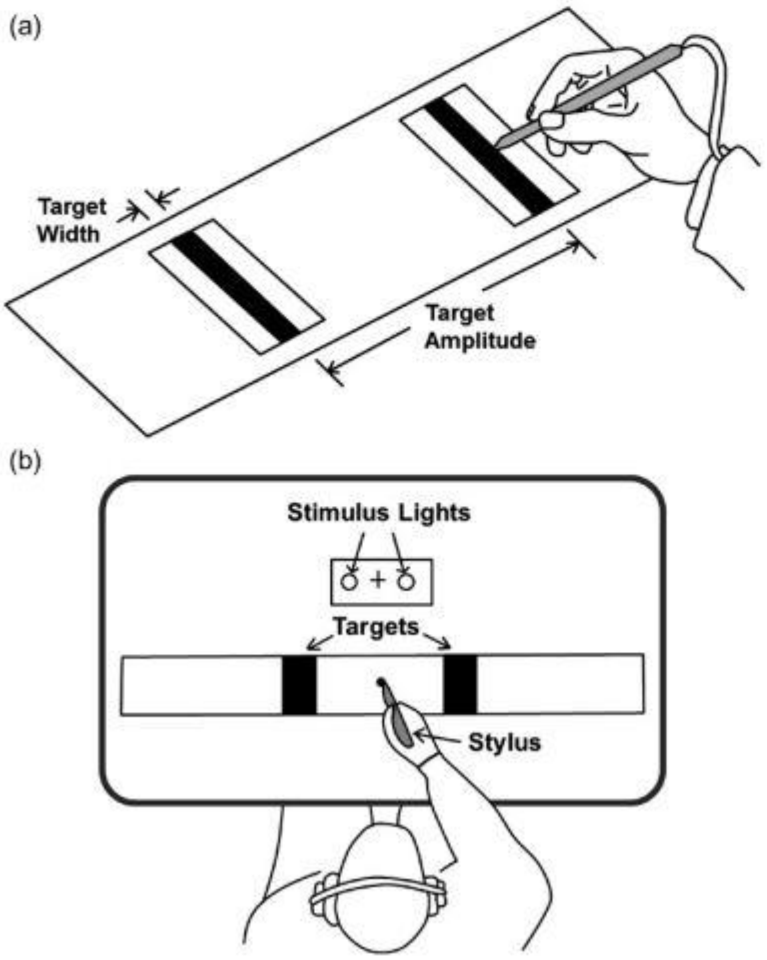


Figure 2: Experimental paradigm for Fitts' law: (a) Serial task. (b) Discrete task. (Note: Figure taken from I. Scott Mackenzie, Human-Computer Interaction an Empirical Research)

The relationship Fitts examined is that between the amplitude of the movement task, the time taken, and the width, or tolerance, of the region within which the move terminates. In what may be the most understated conclusion in all of experimental psychology, Fitts summarized his findings as follows: "The results are sufficiently uniform to indicate that the hypothesized relation between speed, amplitude, and tolerance may be a general one." (Fitts, 1954, p. 389) Indeed, the robustness of Fitts' law is extraordinary. In the decades since Fitts' seminal work, the relationship has been verified countless times and in remarkably diverse settings, whether under a microscope or under water.

Fitts proposed to quantify a movement task's difficulty—ID, the *index of difficulty*—using information theory by the metric "bits." Specifically:

$$ID = \log_2\left(\frac{2A}{W}\right) \tag{3}$$

The amplitude (A) and width (W) in Equation 3 are analogous to Shannon's signal (S) and noise (N) in Equation 2. Note the offsetting influences of A and W in the equation. Doubling the distance to a target has the same effect as halving its size. An important thesis in Fitts' work was that the relationship between task difficulty and the movement time (MT) is linear. The following expression for ID was introduced later to improve the information-theoretic analogy (MacKenzie, 1992):

$$ID = \log_2\left(\frac{A}{W} + 1\right) \tag{4}$$

Because A and W are both measures of distance, the term in parentheses in Equation 4 is unitless. The unit "bits" emerges from the somewhat arbitrary choice of base 2 for the logarithm.

Fitts also proposed to quantify the human rate of information processing in aimed movements using bits per second as the units. This is a provocative idea, based purely on analogy, without a basis in human psychomotor behavior. Fitts' index of performance, now called throughput (TP, in bits/s), is calculated by dividing ID (bits) by the mean movement time, MT (seconds), computed over a block of trials:

$$TP = \frac{ID_e}{MT} \tag{5}$$

The subscript e in ID_e reflects a small but important adjustment, which Fitts endorsed in his 1964 paper. An adjustment for accuracy involves first computing the effective target width as

$$W_e = 4.133\ X\ SD_x \tag{6}$$

Where SD_x the standard deviation in a participant's selection is coordinates. Computed in this manner, W_e includes the spatial variability, or accuracy, in responses. In essence, it captures what a participant actually did, rather than what he or she was asked to do. This adjustment necessitates a similar adjustment to ID, yielding an "effective index of difficulty":

$$ID_e = \log_2\left(\frac{A}{W_e} + 1\right) \tag{7}$$

Calculated using the adjustment for accuracy, TP is a human performance measure that embeds both the speed and accuracy of responses. TP is most useful as a dependent variable in factorial experiments using pointing devices or pointing techniques as independent variables.

If using Fitts' law as a predictive model is the goal, then the movement time (MT) to complete a task is predicted using a simple linear equation:

$$MT = a + b\ X\ ID \tag{8}$$

The slope and intercept coefficients in the prediction equation are determined through empirical tests, typically using linear regression. The tests are undertaken in a controlled experiment using a group of participants and one or more input devices and task conditions.

For an example on how to design and build a Fitts' law predictive model, refer to the book Human-Computer Interaction an Empirical Research Perspective by I. Scott Mackenzie [20]. Fitts' Law has been used in research to model pointing devices by building the prediction equation for estimating the movement time (MT) [15] [21] [19] [22].

In our research work, we built a Fitts' law predictive model for our multimodal input interface to be used to predict the movement time (TM) for targeting objects on a visual interface. We designed a Fitts' law experiment following previous works that we describe in *section 5.1 (Experiment 1: Fitts' law model and* KLM*).*

3.4. The Keystroke-Level Model (KLM)

We equally used the keystroke level model to design a predictive model for our entire multimodal input system. Fitts' law is used to model only the pointing or movement time of our system while KLM is used to provide a predictive model for estimating the total time required to accomplish a task using all or a subset of the features provided by our multimodal input interface.

KLM has widely been used in literature and research. I. Scott Mackenzie in his book Human-Computer Interaction an Empirical Research Perspective [19] describes KLM as follows.

One of the earliest and certainly one of the most comprehensive predictive models in the HCI literature is the keystroke-level model (KLM) by Card, Moran, and Newell. Unlike Fitts' law and the Hick-Hyman law, the KLM was developed specifically for analyzing human performance in interactive computing systems. It was offered as a practical design tool, or as the authors called it, an engineering model. The model predicts expert error-free task completion times using the following elements:

- Task (or series of sub-tasks)
- Method used
- Command language of the system
- Motor skill parameters of the user
- Response time parameters of the system

The model is useful in situations where the sequence of interactions in performing a task is known. For example, if the task is "delete a file," the KLM can predict the time to do the task provided the interactions (operators) can be explicitly specified. If there are two or three different methods to do the task (e.g., mouse + menu selection versus keyboard + command entry), then the KLM can predict the time for each method. If used at the design stage, then design alternatives may be considered and compared.

The KLM is not useful if the details of the interactions are not known. A related model, GOMS (Goals, Operators, Methods, Selection rules), operates at a higher level and attempts, among other things, to predict the method the user will adopt (Card et al., 1983). This is complicated, as it builds upon the cognitive information processing activities of the user and also assumes that users act rationally in attaining their goals.

A KLM prediction requires a task to be broken down into a series of subtasks, or primitive operations. The total predicted time is the sum of the subtask times. The model works with four motor-control operators (K =keystroking, P =pointing, H =homing, D =drawing), one mental operator (M), and one system response operator (R):

$$t_{EXECUTE} = t_K + t_P + t_H + t_D + t_M + t_R \tag{9}$$

Some operations are omitted or repeated, depending on the task. For example, if a keying subtask requires n keystrokes, t_K becomes $n \times t_K$. Each t_K operation is assigned a value according to the skill of the user, with values ranging from t_K = 0.08s for highly skilled typists to t_K = 1.20s for a novice working with an unfamiliar keyboard. The pointing operator, t_P, is a constant based on Fitts' law. The operators and their values from the original KLM are given in table 1 below.

Table 1: : Keystroke-level model (KLM) operators and values. (Note: taken from Human-Computer Interaction an Empirical Perspective by I. Scott Mackenzie)

Operator	**Description**	**Time (s)**
K	PRESS A KEY OR BUTTON Pressing a modifier key (e.g., shift) counts as a separate operation, Time varies with typing skill: Best typist (135 wpm) Good typist (90 wpm) Average skilled typist (55 wpm) Average non-secretary typist (40 wpm) Typing random letters Typing complexcodes Worst typist (unfamiliar with keyboard)	 0.08 0.12 0.20 0.28 0.50 0.75 1.20
P	POINT WITH A MOUSE Empirical value based on Fitts'law. Range from 0.8 to 1.5 seconds. Operator does not include the button click at the end of a pointing operation	1.10
H	HOME HAND(S) ON KEYBOARD OR OTHER DEVICE	0.40
D($n_D . l_D$)	DRAW n_D STRAIGHT-LINE SEGMENTS OF TOTAL LENGTH l_D. Drawing with the mouse constrained to a grid.	$0.9\, n_D + 0.16\, l_D$
M	MENTALLYPREPARE	
R(t)	RESPONSE BY SYSTEM Different commands require different response times. Counted only if the user must wait	t

To validate the KLM, Card, Moran, and Newell conducted an experiment using 14 tasks performed using various methods. Task T1, for example, was "Replace one 5-letter word with another (one line from previous task)." It was performed on three different systems, each with a unique command set. On one system, POET, the required sequence of subtasks was as follows:

Jump to next line	**M K**[linefeed]
Issue Substitute command	**M K**[S]
Type new word	**5K**[word]
Terminate new word	**M K**[return]
Type old word	**5K**[word]
Terminate old word	**M K**[return]
Terminate command	**K**[return]

The operators for each subtask are shown on the right. The task required four mental operations (M) and 15 keystroking operations (K):

$$t_{EXECUTE} = 4 \, x \, t_M + 15 \, x \, t_K \quad (10)$$

t_M was set to 1.35s (Figure 3). t_K was set to 0.23s, based on the mean keystroking time of the participants, determined in a five-minute pre-test. The predicted execution time for task T1 on POET was therefore

$$t_{EXECUTE} = 4 \, x \, 1.35 + 15 \, x \, 0.23 = 8.85s \quad (11)$$

In the experiment, twelve users performed the task an average of 27 times each. The mean execution time was 7.8 s (SE = 0.9 s). So the observation deviated from the prediction by ≈11%. The results were similar for the other tasks. Figure 4 below shows the observed versus predicted task execution times for all 14 tasks (32 tasks, counting method variations). The coalescing of points about the diagonal is clear. This provides a general validation of the model. An arrow identifies the result for task T1 discussed above.

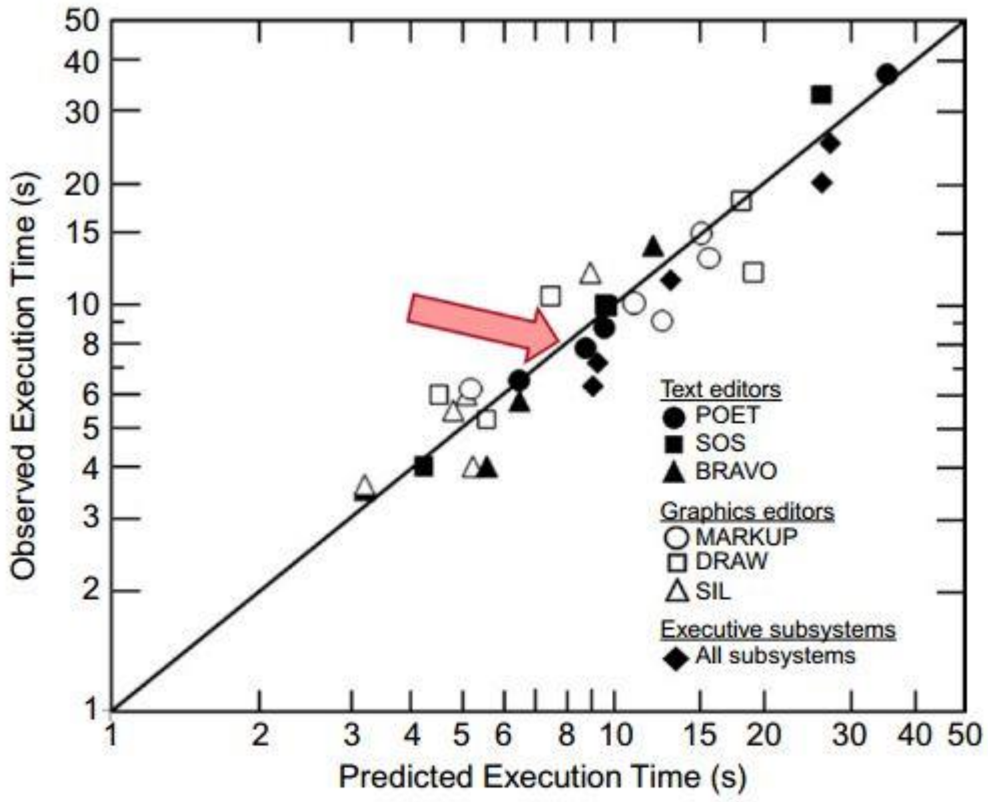


Figure 3: Observed versus predicted execution times for tasks in the KLM validation experiment. The arrow shows task T1 with POET. (Note: taken from Human-Computer Interaction an Empirical Perspective by I. Scott Mackenzie)

For a more detail discussion on KLM and its application, please refer to [19].

In the experimental modelling and evaluation of our system, we built a KLM model and used this model to estimate the task completion time of two experiments that we conducted to evaluate the performance of our system (*Section 5.2*).

3.5. Analysis of variance (ANOVA)

One of our objectives involves the evaluation of the performance of our multimodal input interface on two most common real world browsing scenarios and to compare this with the performance of the mouse and the voice-only input methods. To determine the level of significance of the difference in performance between these three methods, notably the difference in task completion time, we performed an Analysis of Variance test. Details about the experiment are given in the experimental evaluation section (*section 5.2*). What follows is a brief description of ANOVA as summarised from the book Human-Computer Interaction an Empirical Research Perspective by I. Scott Mackenzie [19].

An ANOVA, or *F*-test, is the main statistical procedure for hypothesis testing in factorial experiments. The majority of HCI research papers that describe experiments include the results of an ANOVA, giving an *F*-statistic, the degrees of freedom for the *F*-statistic, and the associated *p* value (explained later). The ANOVA result is typically reported in parentheses as support for a statement indicating whether or not the outcome of the test was statistically significant.

An ANOVA determines if an independent variable—the test conditions—had a significant impact on a dependent variable—the measured responses. The ANOVA determines this by partitioning the variance in the observations on a dependent variable into components attributable to the variance. The "components attributable" are the test conditions (factors and levels) and the participants. If the variance across levels of a factor is large relative to all the remaining variance (other factors and the participants), it is often possible to conclude that the distinguishing properties of the factor—features in the test conditions—*caused* the observed differences in the measurements on the dependent variable. If so, the factor is said to have had a statistically significant effect on the dependent variable.

Why do we conduct an ANOVA test?

Typically, the goal is to compare two or more interfaces or interaction techniques to determine which is better. By "better" we mean superior performance on one or more dependent variables, such as task completion time, error rate, task re-tries, presses of the backspacekey, target re-entries, and so on. It is interesting that the test is called analysis of variance yet we are more concerned with overall performance:

Does it take less time to complete a task using Method A rather than Method B?

The question above refers to time—the overall, average, or mean time that was observed and measured. The mean is calculated for each test condition over multiple users and often over multiple trials of a representative task. In some situations, multiple trials of a task are administered to investigate learning effects, but often the idea of using multiple trials is just to obtain stable and representative measurements. Nevertheless, it is the difference in the means, not the variances, that interests us. Note that while the above research question is certainly important, the statistical question pursued is not a question at all. It is a statement known as the *null hypothesis*:

> *There is no difference in the mean time to complete a task using*
>
> *Method A vs. Method B.*

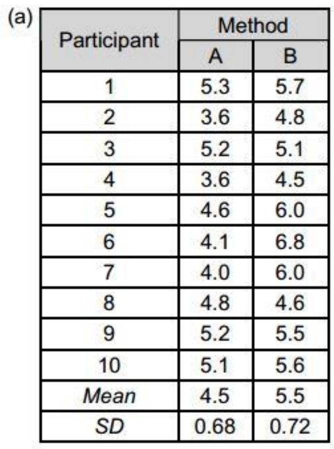
There is an assumption of "no difference," which is a reasonable starting point. The ANOVA tests the data to determine the likelihood of the null hypothesis being true (tenable) or false (rejected). In most cases, the researcher seeks to reject the *null hypothesis*. By convention, a statistically significant result means there is little likelihood the null hypothesis is true. The conclusion in such case is that (a) there is a difference in the means, (b) the difference is statistically significant, and (c) the difference was caused by distinguishable properties in the test conditions. Let's see how this is done.

Consider a hypothetical experiment where 10 participants perform a few repetitions of a task using two interaction techniques. The techniques could be anything; for example, use of a mouse versus a keyboard. A task is chosen that is representative of common interactions, yet likely to expose inherent differences. An example for mouse versus keyboard interaction might be selecting options in a hierarchical menu. For this discussion, we will keep it simple and refer to the techniques as Method A and Method B. Bear in mind that the experiment has a single factor (Interaction Method) with two levels (Method A and Method B).

Performance is measured on a dependent variable such as task completion time (in seconds) and the means are computed for each participant over a few trials. Then the means are computed for each method.

Figure 5(a) below gives an example data from such an experiment as the mean task completion time (in seconds) for each participant for each method along with the overall mean and standard deviation for each method. Figure 5(b) shows the corresponding bar chart embellished with error bars. The error bars show ±1 standard deviation in the means.

(a)



Participant	Method	
	A	B
1	5.3	5.7
2	3.6	4.8
3	5.2	5.1
4	3.6	4.5
5	4.6	6.0
6	4.1	6.8
7	4.0	6.0
8	4.8	4.6
9	5.2	5.5
10	5.1	5.6
Mean	4.5	5.5
SD	0.68	0.72

(b)

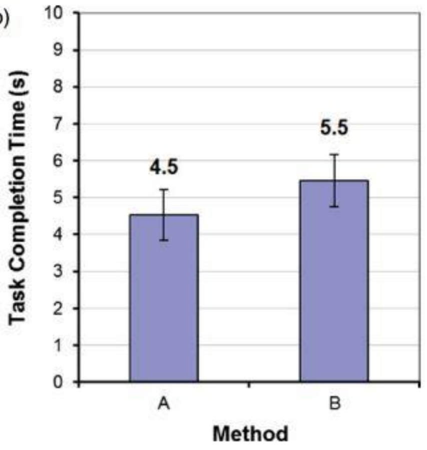
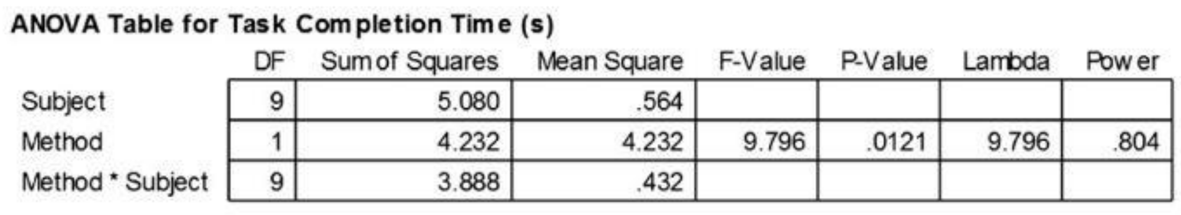


Figure 4:(a)Simulation data . (b) Bar chart with error bars showing ±1 standard. (Note: taken from Human-Computer Interaction an Empirical Perspective by I. Scott Mackenzie)

The step-by-step process of doing an ANOVA on the data in Figure 5a depends on the statistics program used. Nevertheless, the result is a table similar to that shown in the figure below.



ANOVA Table for Task Completion Time (s)

	DF	Sum of Squares	Mean Square	F-Value	P-Value	Lambda	Power
Subject	9	5.080	.564				
Method	1	4.232	4.232	9.796	.0121	9.796	.804
Method * Subject	9	3.888	.432				

Figure 5: Analysis of variance table for data in Figure 5a. (Note: taken from Human-Computer Interaction an Empirical Perspective by I. Scott Mackenzie)

The most important statistic in the table is the “P-Value.” This is the probability of obtaining the observed data if the null hypothesis is true. A low probability indicates that the differences in the means are unlikely to have occurred in view of the null hypothesis. A high probability indicates that the differences in the means may simply be due to chance. With p = .0121, there is a fairly low probability—less than 2 percent—that the difference is simply a chance outcome. In all likelihood, there is some inherent and distinguishing property of one or the other method that caused the difference. Note the term “caused” in the preceding sentence. One reason experimental research is so valuable is that conclusions of a cause-and-effect nature are possible.

It is a convention in experimental research that significance, in a statistical sense, requires pless than .05, or a 1 in 20 probability. In other words, we consider the null hypothesis tenable unless there is a 1 in 20 chance or less that it is not. In addition to the probability, important statistics in the ANOVA are the *F* statistic (F-Value, the ratio of sample variances) and the degrees of freedom for the main effect (Method) and the residual effect (Method * Subject).

An ANOVA test can be designed as within-subjects or between-subjects. In within-subject designs, each participant is tested on all levels of a factor. Meanwhile in between-subjects designs, a separate group of participants is recruited and assigned to each test condition.

For a complete and detailed description of ANOVA and its different forms including tests with more than two test conditions, Post hoc comparisons, Two-way analysis of variance and ANOVA tools, please refer to the book Human-Computer Interaction an Empirical Research Perspective by I. Scott Mackenzie [19].

In this research, we designed a one-way analysis of variance test with three test conditions so as to compare the performance in terms of task completion time of our multimodal input interface with that of the standard mouse and the voice-only input methods on two web browsing case-studies (*Section 5.2*).

4. System Design and Implementation

We have designed and implemented a multimodal input system that makes use of a combination of head movements and voice commands to mimic the operations of the standard mouse. Our system is implemented using a simple RGB camera and an ordinary PC microphone. The RGB camera allows us to track the user's face and use this as input to the multimodal pointing system. Through the microphone, voice commands are captured and sent to the system for further processing and action. Head movements are used as input to control the position of the cursor on the screen. Voice commands on the other hand are captured and used to trigger the different mouse button operations such left button click, right button click, left button double click and mouse wheel scroll in both directions. This multimodal input interface is an alternative to the standard mouse for people with upper limb motor disabilities. The figure below is a conceptual representation of our multimodal input system as an equivalent to the standard mouse. In Figure 7 below, our system is shown to be composed of two main components. A component for controlling the mouse pointer via head movements and a component for execution mouse button actions based on received voice commands.

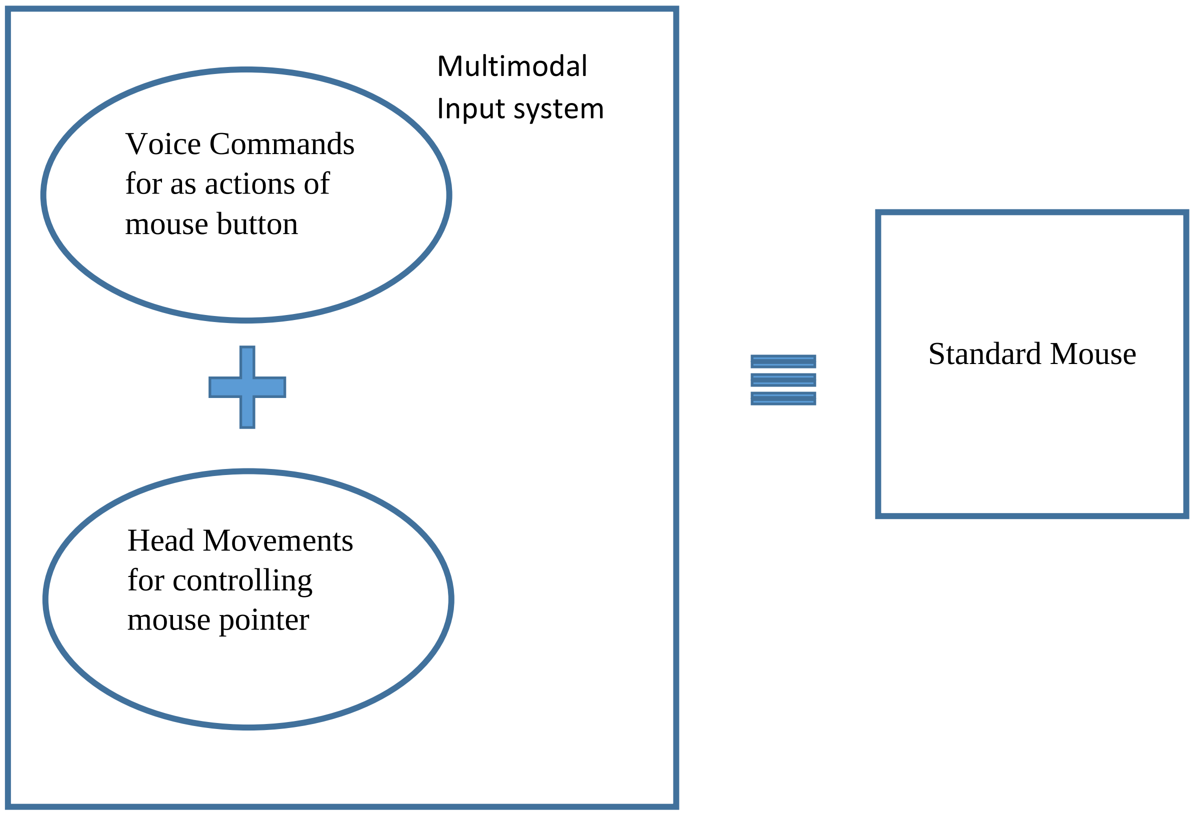

Multimodal Input system
Voice Commands for as actions of mouse button
Head Movements for controlling mouse pointer
Standard Mouse

Figure 6: Conceptual representation of our multimodal input system as an equivalence to the standard mouse

In the sections that follow, we describe the design and implementation of our system in two stages. The first stage explains the tracking of the face to control the cursor on the screen and the second stage talks about the use of voice commands to fire mouse button operations at the current cursor location on the screen. Finally, there is a summary that describes how the two pieces fit together to form the multimodal input interface.

4.1. Controlling the Mouse Pointer Using Head Movements

The tracking of head movements to control the cursor position on the screen follows the steps outlined below.

Step 1: Identify the Screen size

Before reading the first frame from the video camera, the size of the computer screen is identified. Let the screen size be *screenWidth x screenHeight* pixels. Let points on the screen coordinate be identified by the variables p and q, such that a point on this coordinate is denoted *screenLocation(p, q)*, where *0<=p<=screenWidth* and *0<=q<=screenHeight*.

Step 2: Get the size of the video frame

The first video frame is captured from the camera and used to identify the video frame coordinate. This video frame is of size 640 x 480 pixels. We call the coordinate of this frame video-frame. Let points on the coordinate of this frame be identified by the variables (x, y), where, x and y are the corresponding pixel point location such that a point on this coordinate is denoted *videoFrameLocation(x, y)*. The top-left corner point of the video-frame is *videoFrameLocation(0, 0)* and the bottom right corner point is *videoFrameLocation(640, 480)*. The centre of this frame is identified at *videoFrameLocation(320, 240)* pixel point location.

Step 3: Define a virtual mousepad

Around the centre of the video-frame, we define an imaginary/virtual rectangle which we call virtual mousepad. The size of this rectangular mousepad is dynamically set relative to the size of the monitor. Let the size of the virtual mousepad be denoted *padWidth x padHeight* pixels. This is the optimum size of the virtual mousepad determined empirically. An explanation of how this optimum size was determined will be discussed later. The virtual rectangle we have defined acts the same way as the physical mousepad that is found on laptop computers. This is basically where everything is done to control and position the cursor at different locations on the screen. A new coordinate is defined that spans the virtual mousepad. Let points on this coordinate system be identified by variables (u, v), where *0<=u<=padwidth; and 0<=v<=padHeight*. It should be noted that the virtual mousepad is placed at the centre of the coordinates of the video-frame described earlier.

A point, *videoFrameLocation(x,y)*, on the video-frame coordinate obtained by tracking the user's face is normalized to obtain the corresponding point, *virtualPoint(u,v)* on the virtual mousepad coordinate. This newly obtained point, *virtualPoint(u,v)*, on the virtual mousepad coordinate is further processed and used to set the cursor at the corresponding location on the screen. This normalisation process is described next.

The location of the top-left corner point of the rectangular virtual mousepad on the video-frame coordinate is identified and named *zeroPoint(x, y)*. This marks the beginning of the virtual mousepad from the top-left corner on the video-frame. Correspondingly, the location of the bottom-right corner point of the virtual mousepad on the video-frame coordinates is identified and labelled *endPoint(x, y)*. This denotes the end of the virtual mousepad at the bottom right corner on the video-frame coordinates. To normalize a point, say, *videoFrameLocation(x,y)* from the video-frame coordinate to the virtual mousepad coordinate, we subtract the *videoFrameLocation(x,y)* from *zeroPoint(x,y)*. Note that this is a vector subtraction operation where the operation is carried out between corresponding components of the points to produce a new point. The resulting new point, *virtualPoint(u,v)*, corresponds to a location on the virtual mousepad coordinates. If the *u-component* of the resulting point is less than 0, the *u-component* is automatically set to *0*. On the other hand, if the *v-component* of the new point is less than *0*, it is automatically set to *0*. In a similar manner, if the u-component of the point obtained is greater than *padWith*, this u-component is automatically set to *padWidth*. Similarly, if the *v-component* of this new point is greater than *padHeight*, the value of the v-component is automatically set to *padHeight*. By using this process, we convert any point, *videoFrameLocation(x,y)*, on the video-frame coordinate into the corresponding virtual mousepad point, *virtualPoint(u,v)* on the virtual mousepad coordinate and ensure that *u* and *v* span the coordinates *0<=u<=padWdith* and *0<=v<=padHeight* respectively. The figure below shows the first frame read (video-frame coordinate) and the position of the virtual mousepad, its centre point, *zeroPoint* and *endPoint*.

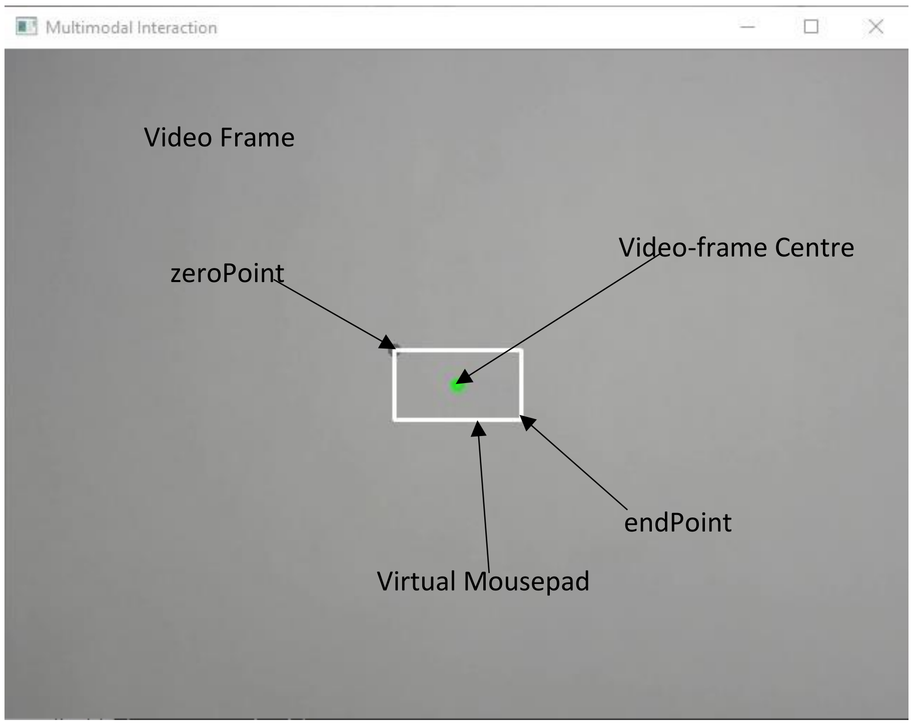


Figure 7: First Frame showing virtual mousepad (white rectangle), centre point (green circle), zeroPoint (top-left corner of virtual mousepad) and endPoint (bottom-right corner of virtual mousepad).

Before reading the first frame as mentioned above, the size of the computer screen is identified. Let the screen size be *screenWidth x screenHeight* pixels. Let points on the screen coordinate be identified by the variables p and q, such that a point on this coordinate is denoted *screenLocation(p, q)*, where *0<=p<=screenWidth* and *0<=q<=screenHeight*. Points on the virtual mousepad coordinate will be translated to corresponding points on the screen coordinate in the process of controlling the mouse pointer through head movement tracking.

Step 4: Track head movements and convert to cursor movements

To track head movements, frames are continuously read from the camera. On each frame read, we run the Viola-Jones algorithm, also known as the Haar Feature-based Cascade classifier to detect a face. Once a face is detected, the rectangular region of the face is identified. Only one face can be detected at a time in our system. This rectangular region of the face is then used for further processing. At first, the centre of the rectangular face region is calculated, let us call this *faceCentre(x, y)* in the video-frame coordinate. It is this face-centre that is used to identify where to position the cursor on the screen. The *faceCentre(x, y)* is normalized to the corresponding point, *virtualPoint(u, v)* on the virtual mousepad coordinate through the process described before. This face-centre plays the same role as a finger does on a laptop mousepad. Here, our mousepad is the virtual rectangle explained previously. The user by moving his/her head around the virtual mousepad can position the face-centre at any point on the virtual mousepad. For each frame read, the identified point location of the face-centre in the virtual mousepad on the video-frame coordinate, *faceCentre(x, y)*, is normalized into the corresponding point, *virtualPoint(u, v)* on the virtual mousepad coordinate as explained above.

The *virtualPoint(u, v)* is then used to place the mouse cursor to a corresponding new location on the screen. Converting a point on the virtual mousepad to the corresponding point on the screen coordinate involves projection. The virtual mousepad is projected to the computer screen. Figure 8 depicts the projection of the virtual mousepad coordinate to the screen coordinate. As an example, a point, A, shown on the virtual mousepad is projected to the corresponding point, B, on the screen coordinate according to equation 12 below.

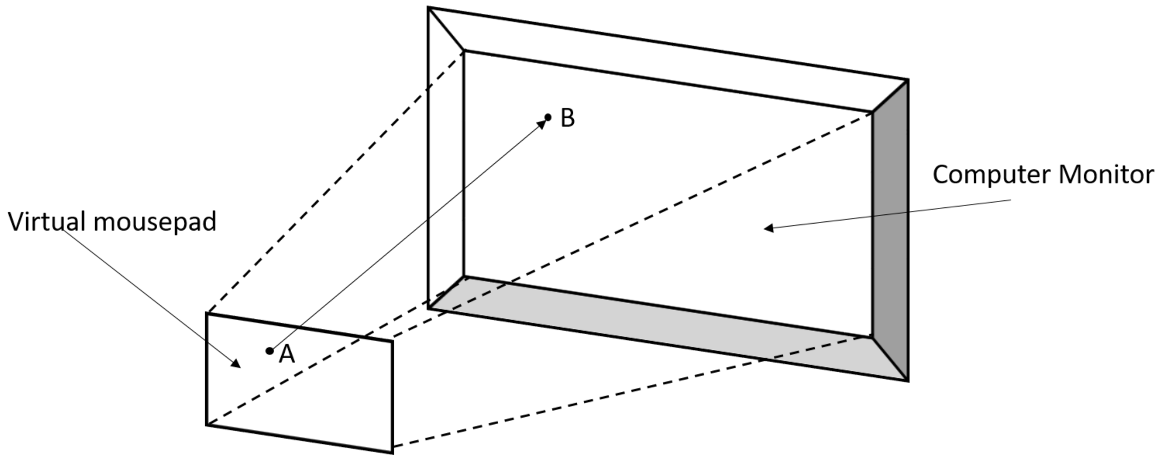


Figure 8: Projection of the virtual mousepad to the screen coordinate. Point A on the virtual mousepad is projected to point B on the screen

Now, with the normalised virtual point, *virtualPoint(u, v)*, to obtain the new position at which to place the mouse pointer at on the screen coordinate, *virtualPoint(u, v)* is projected into the corresponding point *screenLocation(p, q)* on the screen coordinate according to equation 12 below.

$q = (u * screenWidth) / padWidth;$ 12(a)

$q = (v * screenHeight) / padHeight$ 12(a)

This projected point, *screenLocation(p, q)* is then used to set the position of the cursor at location *(p,q)* on the screen. Setting the position of the cursor at point defined by *(p,q)* is accomplished by calling system functions and passing the point *(p,q)* as parameters.

This procedure is repeated for each frame read to move the cursor around the screen as the user moves his/her head (face-centre) around the virtual mouse pad. The figure below is an example showing the position of the user's face, face-centre, and virtual mousepad on a read frame.

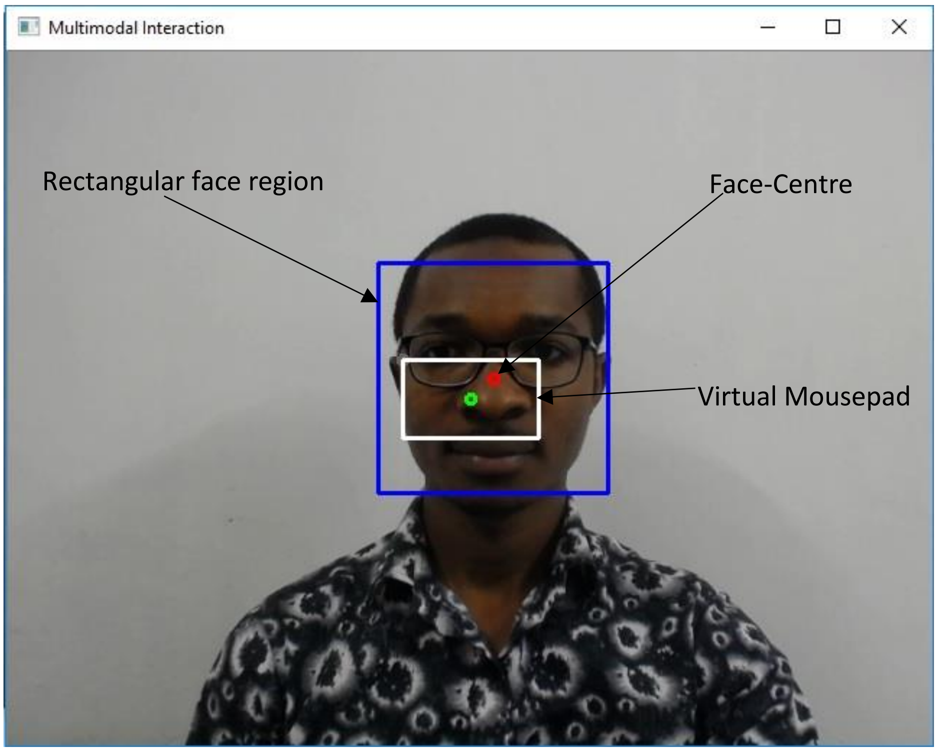


Figure 9: Detected user's face as identified by the blue rectangle and face-centre represented by red circle.

As mentioned above, coordinates of the virtual mousepad are translated to the coordinates of the screen. Accordingly, the *zeroPoint(x, y)* defined above corresponds to *screenLocation(0,0)*, that is the top left corner of the screen. Correspondingly, the *endPoint(x,y)* corresponds to *screenLocation(screenWidth, screenHeight)* which is identical to the bottom right corner of the screen. We call this mechanism of tracking head movements to control the mouse pointer the continuous tracking mode. The next section describes the break-reset tracking mode in which the inertia associated with head movements is controlled.

Break and reset movement with inertia control

To control the inertia due to head movements and to allow more freedom and flexibility to the user while controlling the mouse pointer, we introduced the mechanism of break and reset in the cursor control. This mode of controlling the mouse pointer is called the break-reset tracking mode. We allow the user to reset the position of his/her head in the course of moving the cursor to a target similar to what we will do with a real mouse or physical mousepad where we sometimes pause and reset the mouse position or finger position in the midst of an action. Rather than have the user move his/her head continuously in the direction of the target until the target is reached, we made the control discrete and independent of the location of the user's head. That is, after covering a certain distance on the path to the target, the user can reposition his/her head without affecting the position and distance already covered by the cursor. To achieve this, we defined two states, the 'go' state and the 'break' state. The 'go' state is synonymous to when the finger is placed on a physical mousepad or when a physical mouse is on the mousepad with the hand on top. The 'break' state is identical to the state during which the finger is lifted from the physical mousepad or when the physical mouse is lifted in the air to reset its position. When the user activates the 'go' state, the vector distance covered by the head from the moment he/she entered the 'go' state to when he/she exits the 'go' state is translated to the corresponding vector distance in the screen coordinate and added to the current vector position of the cursor on the screen to determine the new position of the cursor on the screen. In the 'break' state, detected movements of the user's head have no effect on the position of the cursor on the screen. Figure 10 and 11 depict an example of the sequence of steps taken to move the mouse pointer from an initial location, say, P to a final location, S on the screen. Figure 10 shows the initial point and the final point. The mouse pointer is circle in red in both figures.

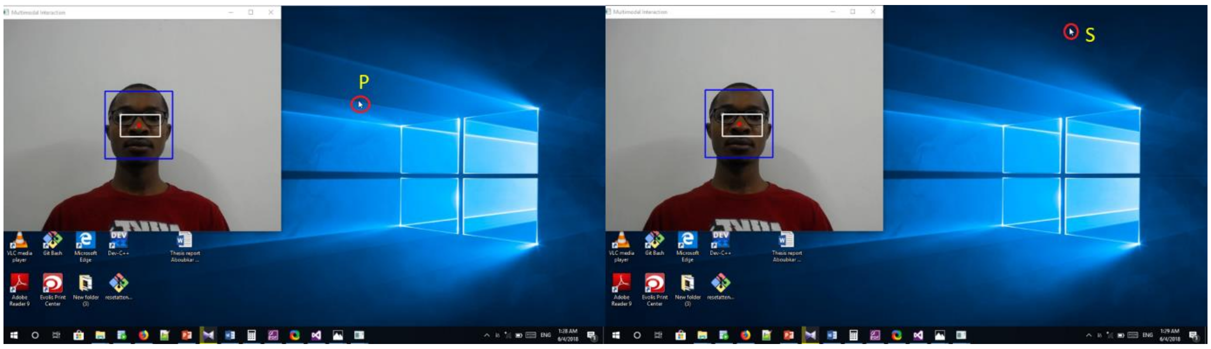


Figure 10: Initial location of the mouse pointer, P, and final location, S

Figure 11 illustrates the steps taken by the user to move the mouse pointer from point P to point S. The user completes the task in three phases. Initially, the user moves the mouse pointer from point P to point Q during the 'go' state (right image of Phase 1). Next, the users enters the 'break' state and repositions his head (left image of Phase 2). The user activates the 'go' state again and moves the mouse pointer to point R (right image of Phase 2). The user again enters the 'break' state, resets the position of his head (left image of Phase 3) and finally enters the 'go' state again and takes the mouse pointer to the point S (right image of Phase 3). Notice that the movement to reset the position of the head does not affect the current position and distance already covered by the mouse pointer.

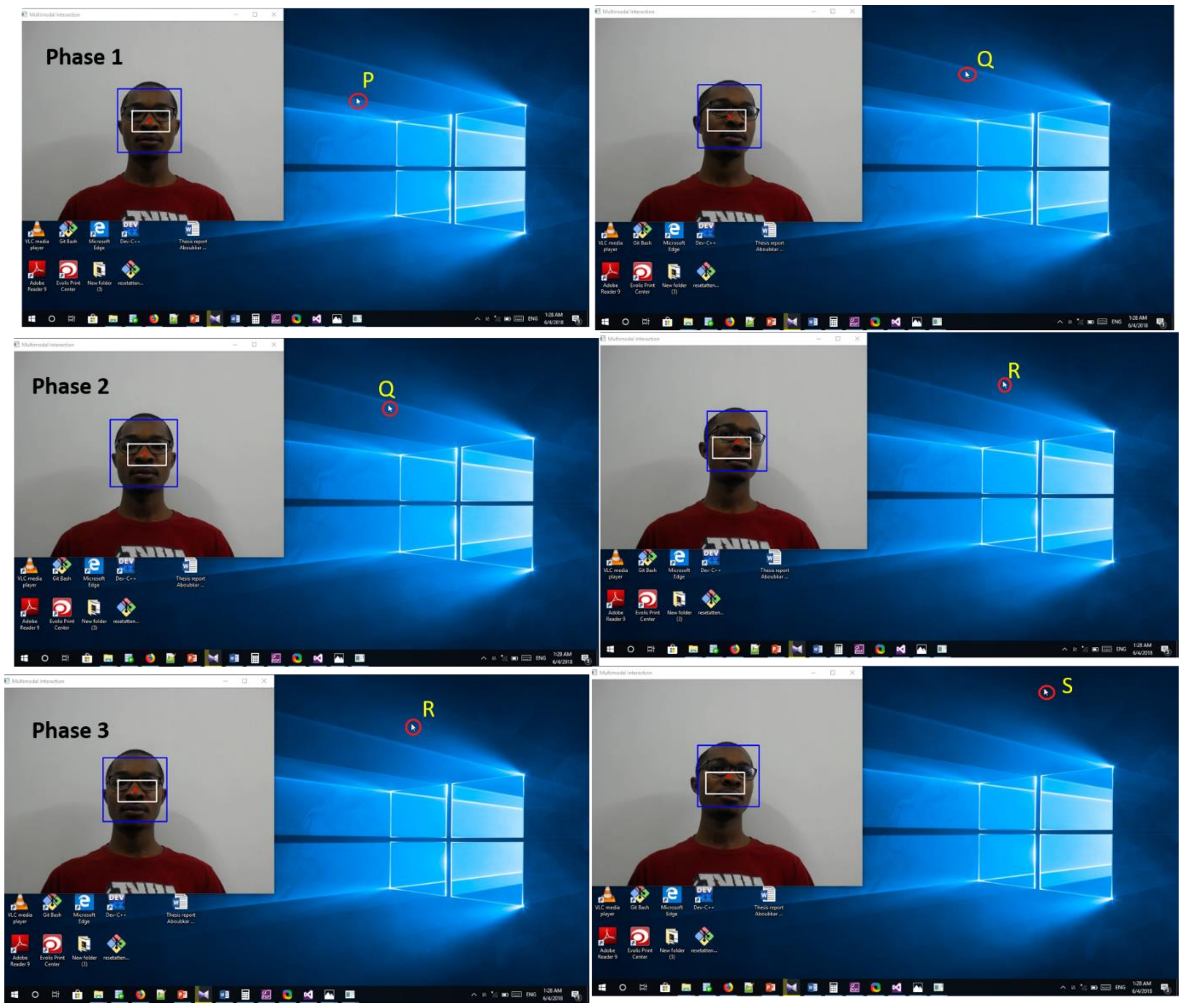


Figure 11: Sequence of break and reset steps to move the mouse pointer from location P to location S

To prevent unwanted cursor movements when a user is simply seating in front of the computer but with no intention to move the cursor, we defined two global states: 'active' and 'inactive". In the 'active' state, detected head movements are used as input to control the mouse pointer. In the inactive state, the cursor on the screen is not affected by detected head movements. State activation and deactivation is done through voice commands.

As mentioned above, we have used the Viola-Jones algorithm for detecting the human face. This is used only as a pre-processing step and the limitation of the Viola-Jones algorithm in terms of the angle of detection of the face will have little effect on the proper functioning of the system. The design of the virtual mousepad and the requirement for the user to face the screen by seating in front of the camera guarantees that the left to right rotation of the face will not exceed an angle of 15 degrees. Rotating the face above 15 degrees on either direction will have ergonomic consequences on the part of the user. This can lead to neck pain. The break-reset tracking mode is specifically designed to prevent stressful movement from the part of the user.

To summarize, each time a frame is read, the location of the centre point of the rectangular face region of the user on the virtual mousepad is calculated. This point is translated to the corresponding point in the screen coordinate and used to set the new location of the mouse pointer on the screen. In the case break movements to reach target, the vector distance covered by the face-centre is translated and added to the current vector position of the mouse pointer on the screen and consequently set the new location of the mouse pointer.

In order to smoothen and stabilize the cursor movement on the screen, a Kalman filter is applied to predict and optimize the target points as frames are read from the camera and processed.

4.2. Triggering Mouse Button Actions Using Voice Commands

The second part of our system involves using voice commands to trigger mouse button actions and other mouse related functions. This component demands the use of a speech recognition engine for identifying and processing the voice commands. The speech recognition engine will detect and parse the voice commands echoed by the users. A parsed command will be converted into text. The identified text can then be used to perform a desired action as defined by the application. The first thing we did to implement the voice component of our system was to identify a set of possible words or short phrases that can be used as voice commands to mimic the different mouse button actions such as left button click, right button click, double click, etc. After interviewing a pools of users, we identified the following short and simple English words/phrases. Each word/phrase corresponds to a mouse button action. The words/phrases are chosen such that they are easy to pronounce and to remember without difficulty by the vast majority of users. In fact, users with previous computer knowledge will find it easy to use and to remember because these phrases are common and frequent among computer users. Table 2 below shows the list of words/phrases. Note that words in parentheses as shown on table 2 are optional. That is a user can choose to omit the word in parenthesis and the voice command will remain the same. For example, the voice command "Page Up" can optional be invoked by simple uttering the word "Up".

Table 2: List of voice commands used in our multimodal input system

Word/Phrase	**Corresponding mouse button action**	**Description**
Left click	Left mouse button click	
Right click	Right mouse button click	
Double click	Double clicking on left mouse button	
Drag	Left mouse button press	
Drop	Left mouse button release	
(Page) Up	Mouse wheel scroll towards user	
(Page) Down	Mouse wheel scroll away from user	
Go		To enter the 'go' state
break		To enter the break state
start		To enter the active state
stop		To enter the inactive state

From the table above, the voice command "left click" corresponds to performing a left button click on the standard mouse. When the identified voice command will correspond to the phrase “left click”, the system will automatically fire a left mouse button click action at the current location of the cursor on the screen. A similar process will occur for the “right click”, “double click”, “drag” and “drop” voice commands. The phrases “page up” and “page down” correspond to the action of scrolling the mouse wheel to move the content of a page up and down respectively.

With the list of possible voice commands identified and established, the next step is to implement a speech recognition engine that will recognize our voice commands and perform a desired action following an identified command. At first, the voice commands are defined and stored in an xml document. This is then passed to our speech recognition engine at runtime. The speech recognition engine uses this as a database to identify and match incoming phrases or words uttered by a user. Once a word or phrase is successfully recognized and matched with a word or phrase in the database, its corresponding action is executed by the system. For each of the voice commands, a callback function is implemented to execute a predefined action. This function fires when the associated voice command is received and parsed by the system.

To put the two pieces together to create the multimodal input interface, the two components are implemented to run in parallel. While the system tracks a user’s head and uses this as input to control the mouse pointer on the screen, it simultaneously listens for voice commands from the user through its speech recognition module. The action tied to each voice command is executed on the cursor at its present location as determined by the tracked head movements. This multitasking system constitutes the multimodal input interface that functions as an alternative to the standard mouse.

In summary, the entire multimodal interface makes use of head movements and voice commands and is designed to function as a pointing device. The system is made up of two threads. One thread is responsible for capturing the user’s head movements and translating this into the movement of the mouse pointer on the screen. The second thread on the other hand listens to voice commands from the user. Each voice command corresponds to a mouse button or mouse wheel action. Once a voice command is recognized by the system, the corresponding action is performed on the mouse cursor at the current location on the screen as determined by the head movements. This combination of head movements and voice commands constitutes our multimodal input system.

The entire system is implemented using the C++ programming language on Microsoft Visual Studio 2015. All the computer vision and image processing steps required to track the head movements are accomplished using the OpenCV library functions. Windows library functions are used to set the cursor position on the screen. The speech recognition part is implemented using the Microsoft Speech API (SAPI). SAPI provides capabilities that allow applications to use speech recognition and speech synthesis. The system was developed on an HP (Hewlett Packard) laptop computer with Intel processor Intel® Core™ i5-6200U CPU @ 2.30GHz 2.40GHz.

5. Experiment and Experimental Result

To evaluate our system, we designed and conducted two experiments. The first experiment consisted in establishing a keystroke-level model (KLM) for the prediction of the task completion time of a given task using our multimodal input interface. Part of this involved designing a Fitts' law experiment to construct a prediction equation for predicting the pointing time i.e. the time required to reach a target using head movements in our multimodal input interface. The second experiment is a comparative evaluation of our multimodal input interface in web navigation by making use of two real world scenarios. We aimed at finding out how well our system will do on web navigation as compared to the standard mouse and voice-only navigation methods. The mouse used in all experiments in this work is an A4 Tech OP-620D USB Optical Mouse.

5.1. **Experiment I)** Fitts' Law Model and KLM Design

The purpose of building the Keystroke-Level model (KLM) for our system is to provide a tool for designers to be able to theoretically predict or estimate the time required to complete a task using our multimodal input interface without actually diving into the implementation of the task. Given two or more pointing interfaces, each with its own KLM model, a designer can easily compare the task completion times of these interfaces on a given task by simply calculating these time through the respective KLM predictive models. This saves the time to actually conduct the task with each of the interfaces to determine the task completion times. In order to construct the KLM model of a pointing interface, we need a prediction model for the pointing time for that interface. This equation to predict the pointing time is usually defined through a Fitt's law experiment. Therefore, providing these two models will make the design and implementation of our interface complete.

5.1.1. Fitts' Law Model for Pointing Time

Fitts' law is generally used to model interaction techniques that involve rapid-aimed movement to target and select objects. An example is moving a mouse cursor to select an object on a screen. We use Fitt's law in this experiment to build a prediction equation for the pointing time of our multimodal input interface and determine whether it conforms to the model. We also use this Fitts' law experiment to compare the performance of our multimodal input interface with that of the standard mouse by using the resulting throughput. The resulting prediction equation is also used to build a complete KLM model for our system.

We designed a series of target selection tasks similar to previous Fitts' law experiments. We defined 3 different amplitude (A) i.e 80, 160 and 300 pixels and 3 different widths (W) namely 20, 40, 60 pixels. A cross of the amplitudes and widths gave a total of 9 target conditions. For each target width, a participant performed a centre-out task at 3 different locations on the screen representing the three defined amplitudes. The figure below shows each width and amplitude combinations.

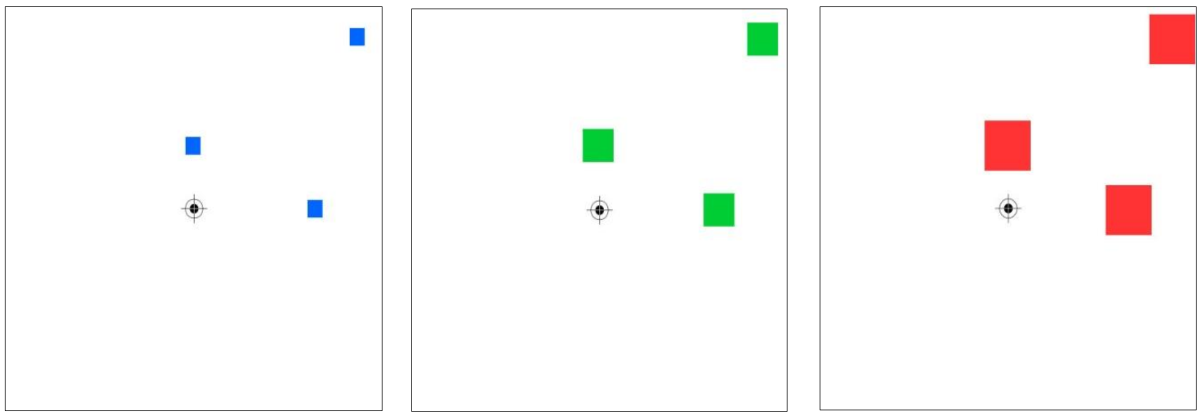


Figure 12: Targets of 20 (blue), 40 (green) and 60 (red) pixels at Amplitudes of 80 pixels (vertical), 160 pixels (horizontal) and 300 pixels (diagonal)

Each participant was asked to seat in front of the monitor at normal and comfortable browsing distance. When a participant was ready, targets were presented one at a time and the participant was asked to move the mouse cursor from the centre of the image to the centre of the target as fast as he/she could using the multimodal input system. The time taken to move the cursor from the centre of the image to a target was recorded. For each target, the participant performed multiple trials and the average time was taken. A total of 10 participants performed the experiment. To capture what each participant actually did, we calculated the effective target width (We) using SD_X for each target selection (see section 3.3 for more on this).

$$We = 4.133\ X\ SD_X \qquad (13)$$

Where SD_X is the standard deviation of x and x is the distance between the participant's pointed spots in reaching the target to the center of the target.

The same experiment was performed, but this time using the standard mouse so as to act as a baseline for our system.

Table 3 below shows the data for the multimodal input system and the mouse. The table presents the nine A-W combinations and the data calculated from the participant responses. We is the effective target width as defined previously. ID_e is the effective index of difficulty. MT is the mean movement time in milliseconds. And TP is the throughput in bits/s

$$ID_e = \log_2\left(\frac{A}{W_e} + 1\right) \quad (14)$$

$$TP = \frac{ID_e}{MT} \quad (15)$$

Finally, a Fitts' law prediction equation to predict the pointing time was built for the multimodal input interface and as well as for the mouse using the data shown in table 3 below. Our prediction equation uses MT as the dependent variable and ID_e as the independent variable. As explained in the prerequisite section (Chapter 3), ID_e is used because it includes both the speed and accuracy of user responses. It models what the user actually did rather than what the user was asked to do.

Table 3: Fitts' law experiment results for the mouse and our multimodal input interface

A (pixels)	**W (pixels)**	**ID (bits)**	**Mouse**				**Multimodal Interface**			
			We (pixels)	**IDe (bits)**	**MT (ms)**	**TP (bits/s)**	**We (pixels)**	**IDe (bits)**	**MT (ms)**	**TP (bits/s)**
80	20	2.32	17.23	2.50	716	3.49	22.39	2.19	1163	1.89
80	40	1.58	37.97	1.64	595	2.75	41.36	1.55	943.3	1.65
80	60	1.22	59.53	1.23	513	2.40	55.53	1.29	854.6	1.51
160	20	3.17	18.59	3.264	908	3.60	19.29	3.22	1374	2.34
160	40	2.32	35.56	2.46	806	3.05	38.94	2.35	1155	2.04
160	60	1.87	57.04	1.93	684	2.82	57.63	1.92	1031	1.86
300	20	4	22.99	3.81	1027	3.71	20.97	3.94	1898	2.07
300	40	3.09	44.98	2.94	997	2.95	37.60	3.17	1675	1.89
300	60	2.58	61.93	2.55	845	3.019	63.34	2.52	1478	1.71
	Average	2.46	39.54	2.48	789	3.09	39.67	2.46	1286	1.88

Figure 11 below shows the scatter plots and regression lines with associated equations for both methods using the data in table 3.

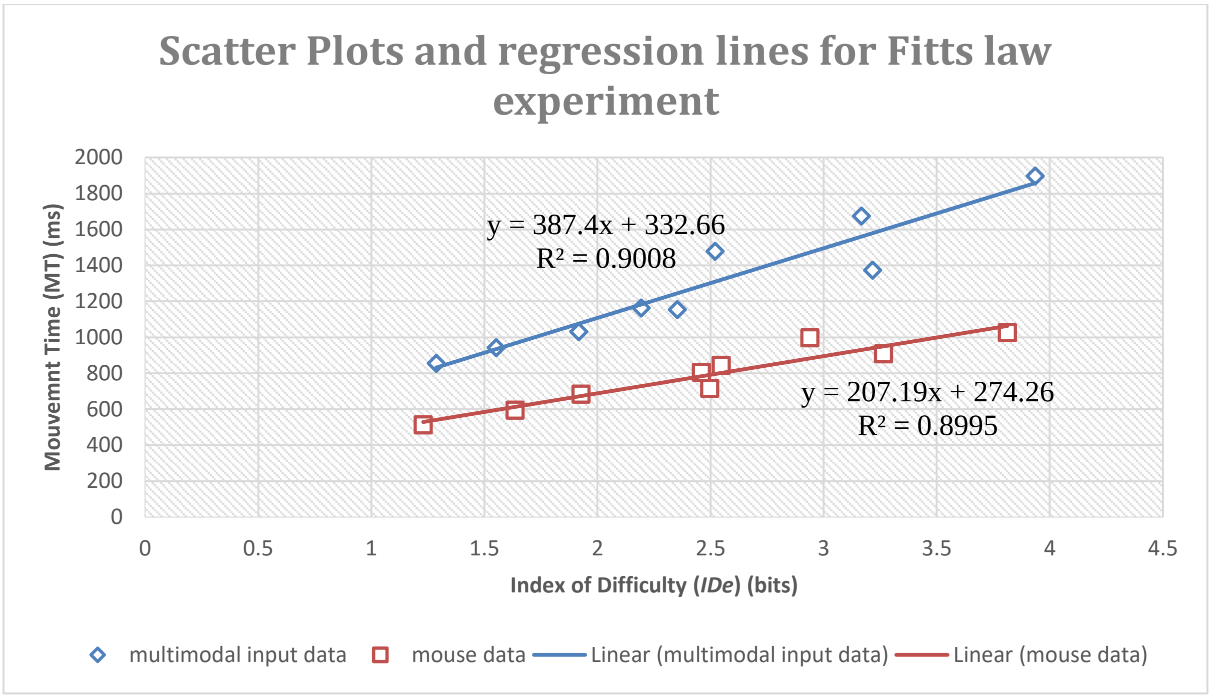


Figure 13: Scatter plots and regression line for our Fitts' law experiment

After performing the experiment and processing the collected data and subsequently drawing the scatter plots and finding the regression lines as shown in Figure 11, we obtained the following prediction equation for our multimodal input interface. This equation predicts the movement time or pointing time.

$$MT = 333 + 387\ x\ ID \tag{16}$$

In order words, using our multimodal input system, the predicted time to move the cursor to select a target of width W over a distance of A pixels is given by the equation:

$$MT = 333 + 387\ x\ \log_2\left(\frac{A}{W} + 1\right)\ ms \tag{17}$$

Therefore by using the equation above, designers can predict the movement time in milliseconds needed to move the mouse pointer on the screen to select targets of any size and at any distance from the point of origin. This allows designers to easily make choices between interfaces at design time without going much into the details of implementation.

The prediction equation for the mouse using the data from our experiment is given below as can be seen on Figure 11 above.

$$MT = 274 + 207\ x\ ID \tag{18}$$

This is equivalent to the following equation in terms of target width, W and distance to target, A.

$$MT = 274 + 207 \; x \; \log_2 \left(\frac{A}{W} + 1\right) \; ms \qquad (19)$$

By analysing the plots and the data on Figure 11, we can conclude that point-select operations using the mouse and our multimodal input system conforms to Fitts' law because of the high R^2 values observed on the plots.

It is also clear from the graph that our multimodal input interface is slower than the mouse in selecting targets of the same width at the same distance. For all target and width combinations, the movement time (MT) for the mouse was lower than the movement time (MT) for the multimodal input interface.

The throughput (TP) for the mouse is 3.09 bits/s higher than the throughput (TP) for the multimodal input interface which is 1.88 bits/s.

To summarize, in this experiment, we have built a prediction equation for our multimodal input interface. This equation allows us to estimate the time required to move the mouse pointer to a target having width W pixels and at the distance of A pixels from the starting point. This time is referred to as the movement time or pointing time. We equally built a prediction equation for the mouse using the same experimental design and setup to act as a basis for our system. We found out that the mouse is faster than our multimodal input interface. However, the speed, accuracy and throughput (TP) of our interface suggest that it can be used and especially by people with upper limb motor disability as an alternative to the standard mouse.

5.1.2. KLM model for our multimodal input interface

The second part of this experiment was to construct a complete KLM model for the multimodal input interface that provides completion time estimates for pointing (as determined by Fitts law) as well as for voice commands. This model can then be used to estimate task completion times for a complete interaction task that uses the multimodal input system. An example of this estimation process is given in the next section.

We set up a simple experiment to determine the average time taken to utter each of the voice commands presented in *table 1*. We had 12 participants. Each participant performed the experiment in two modes: the slow pronunciation mode where the user uttered the voice command slowly such that the uttered command was recognizable by the system and the fast pronunciation mode in which the user uttered the voice commands as fast as he/she could while making sure that the intended command was recognizable by the system. The average time taken for each voice command per mode was calculated. The results are summarized in the table below.

Table 4: Average time in seconds taken to utter each voice command in slow mode and fast mode

Voice command	Pronunciation time in seconds	
	Fast pronunciation mode	Slow pronunciation mode
left click	0.91	1.6
right click	1.00	1.67
double click	0.92	1.51
drag	0.82	1.31
drop	0.73	1.17
page up	0.95	1.36
page down	1.05	1.5
Average time	**0.91**	**1.45**

Using the above data and the results of the Fitts' law experiment as well as some referenced value, we built the KLM model for our system as shown in table 4 below. For a task that involves one or more of the operators shown in table 4, the time to complete the task can be estimated by adding the time associated with each operator. How the estimation is performed using KLM operators is explained in chapter 3. We have demonstrated this estimation process in the next sections.

Table 5: KLM operators and task completion time values for our multimodal input interface

Operator	**Description**	**Time (s)**
V	ECHOEING A VOICE COMMAND This is to produce a mouse button effect. Includes "Left Click", "Right Click", "Double Click", "Drag", "Drop", "Page Up", "Page Down" Timing depends on user Fast pronouncer: Slow Pronouncer:	 0.91 1.45
P	POINTING USING THE HEAD Empirical value based on Fitts' law.	$333 + 387\, x\, \log_2\left(\frac{A}{W} + 1\right)\ ms$
M	MENTALLY PREPARE	1.35 (reference value)
R(t)	RESPONSE BY SYSTEM Response time differs by commands	t

Therefore, given a task that can be accomplished using a combination of one or more operators from the table above, the task completion time can be estimated using the value for each of the operator from the table. It should be noted that this is just an estimate. The response time by the system R(t) depends on the command and the application domain in question. The mental operator (M) is the time the user takes to think on what to do next. This value on the table is a reference from the original KLM experiment as explained in [19].

5.2. **Experiment II)** Comparative Evaluation of our system in web navigation

In this experiment, we designed two tasks based on common real-world every day browsing scenarios and measured the time to complete each task using a mouse, voice-only input and our multimodal interaction method. Our objective was to evaluate the performance of our system in web browsing and compare this with the standard mouse-controlled web browsing and the unimodal voice-controlled web browsing. Put in other words, we performed an analysis of variance (ANOVA) test for each task so as to compare the performance of the three methods mentioned, namely the mouse, voice-only input, and multimodal interface of voice commands and head movements.

Note that because we could not lay hands on a voice-controlled web browser, we use the standard Wizard of OZ method to mimic a voice-controlled web browser for the purpose of these tests. The multimodal input interface is used in the continuous tracking mode in these experiments.

This experiment consisted of two sub-experiments. The first experiment required participants to complete a typical e-commerce browsing task on an e-commerce website. The example website used was Amazon.com. The second experiment involved a social networking website. Again, participants had to accomplish a task using the three different methods mentioned above on a popular social networking website. In this case, Facebook.com was used to conduct the experiment. It should be noted that each experiment is a within-subject one factor with three levels analysis of variance (ANOVA). Within-subject because each participant performed each test using all levels of the factor (interaction methods).

For these two experiments, we had a total of 12 participants. Among these 12 participants, 8 of them had experience using our multimodal input method as they took part in the first experiment described earlier (section 5.1). The remaining 4 participants were experiencing the multimodal input system for their first time. Before the beginning of each experiment, participants were briefed on the tasks they were expected to accomplish. A demo was presented to the participants. Participants where free to ask any questions related to the experiment. Participants too had the liberty to continue the experiment or to end it at any time if they felt they needed to do so. Before starting the actual test, each participant was given a series of trials to experiment with and to acquaint with the system.

5.2.1. Experiment II-A: Comparison of task completion time on a typical e-commerce website. Case-Study: Amazon.com

In this experiment, users had to perform a typical task common in an e-commerce website. We designed a simple task that involved successfully adding an item to a shopping cart on the Amazon.com website. The task was designed so as to test the performance of our multimodal input system on different attributes of such a website including selecting an item from a hierarchical menu, selecting a link from a list of hyperlinks, scrolling down a page, clicking a button etc. As a user performed the task using each input method, the time was recorded.

The task involved the following subtasks or steps:

1) Selecting a top-level menu item, which defined what the user wanted to do.
2) Select a category of products from which to shop from a drop down list
3) Select a sub-category of products from a list of hyperlinks
4) Scroll down to a group of products to buy from
5) Select a group of product identified by a hyperlinked image
6) Select the first item shown from the list of displayed items
7) Add the item to the cart by clicking a button
8) Finally proceed to checkout by clicking a button

Each participant had to perform the above task using each of the three input methods mentioned. This is the within-subject test as explained earlier.

Using the designed KML model of our multimodal input system explained in section 5.1 we estimated the time to complete this task as follows.

Table 6: Subtasks involved in experiment II-A with associated KLM operators for our Multimodal input interface

Serial No.	Sub Task	Operation	t_P (s)
1	Move mouse over target top level menu	MP[80, 21]	1.23
2	Hover mouse pointer over product category	MP[345, 21]	1.95
3	Select sub-category	MP[348, 21]V	1.96
4	Scroll to group of product	V	0
5	Click on hyperlinked image	MP[361, 100] V	1.18
6	Select item	MP[120, 226]V	0.57
7	Click Add button	MP[753, 206]V	1.19
8	Click checkout button	MP[179, 40]V	1.28
		Σt_P	9.36

The estimated task completion time,

$$t_{Execute} = 7t_M + \sum t_P + \ 6t_V + R(t) \tag{20}$$

Where $R(t)$ is the total system response time that depends on the internet speed and the system.

In order to get a better and more accurate estimate, we calculated A and W for the target at each step and calculated t_P using the Fitts' law prediction equation of our multimodal input interface (Section 5.1). In the table above, P is denoted as P[A, W], where A is the distance to target and W is the target width. By using our Fitts' law equation, we calculated P for each step as follows:

$$t_P = 333 + 387\ x\ \log_2\left(\frac{A}{W} + 1\right)\ ms \tag{21}$$

Taking $t_V = 0.91s$ for the fast pronunciation mode as shown on table 4, the estimated time to complete the task is

$$t_{Execute} = 7(1.35) + 9.36 + \ 6(0.91) + R(t) \tag{22}$$

$$t_{Execute} = \big(24.27 + R(t)\big)\ s \tag{23}$$

Therefore, the estimated time to complete the task using the multimodal input system in fast pronunciation mode is 24.27s plus the system responds time which depends on the system and internet speed.

In the slow pronunciation mode, $t_V = 1.45$, the estimated time is

$$t_{Execute} = 7(1.35) + 9.36 + \ 6(1.45) + R(t) \tag{24}$$

$$t_{Execute} = \big(27.51 + R(t)\big)\ s \tag{25}$$

The estimated time to complete the task considering the slow pronunciation mode is 27.52s plus the system responds time.

Therefore, using our KML model designed in section 5.1, we have estimated the time it will take to complete the task using the multimodal input system. The model allows us to make design decisions and to compare methods without actually running the tasks. Thus useful in saving time and obtaining better insight about the problem.

Figure 11 and figure 12 show the screenshots and the targets to select at each stage of the task as recorded at the time of execution of this experiment on the Amazon.com website. The targets to select at each step is circled in red. The time taken to complete the task, using each method by participants was recorded.

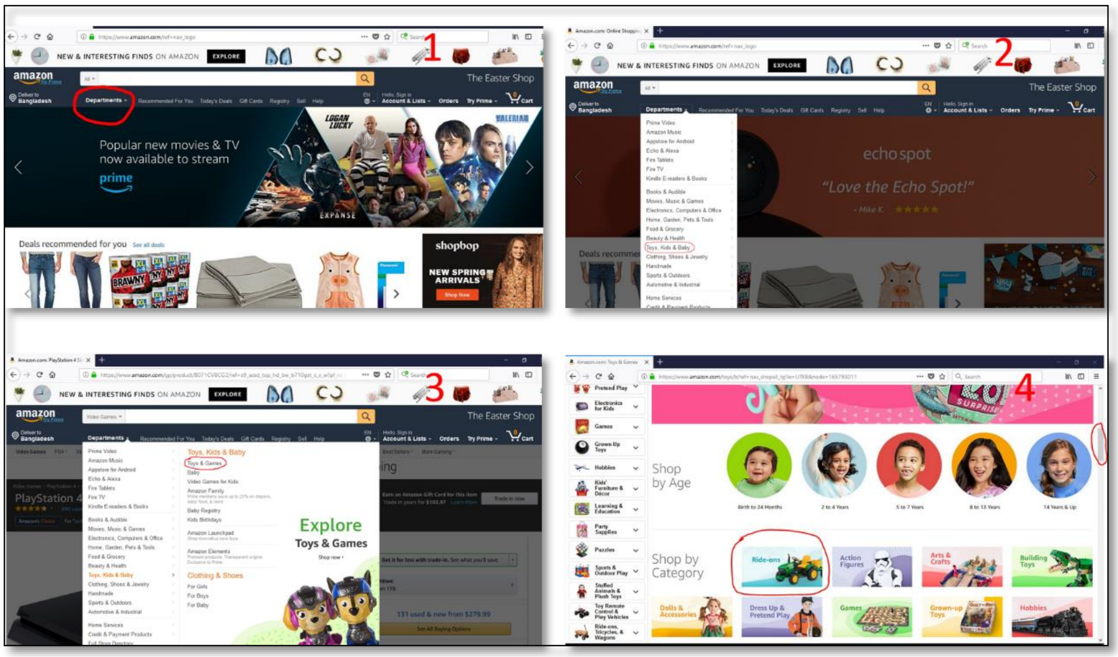


Figure 14: sequence of targets to select for subtasks 1 to 5 in the e-commerce website experiment

Screenshot 4 in Figure 11 contains both subtask 4 and 5.

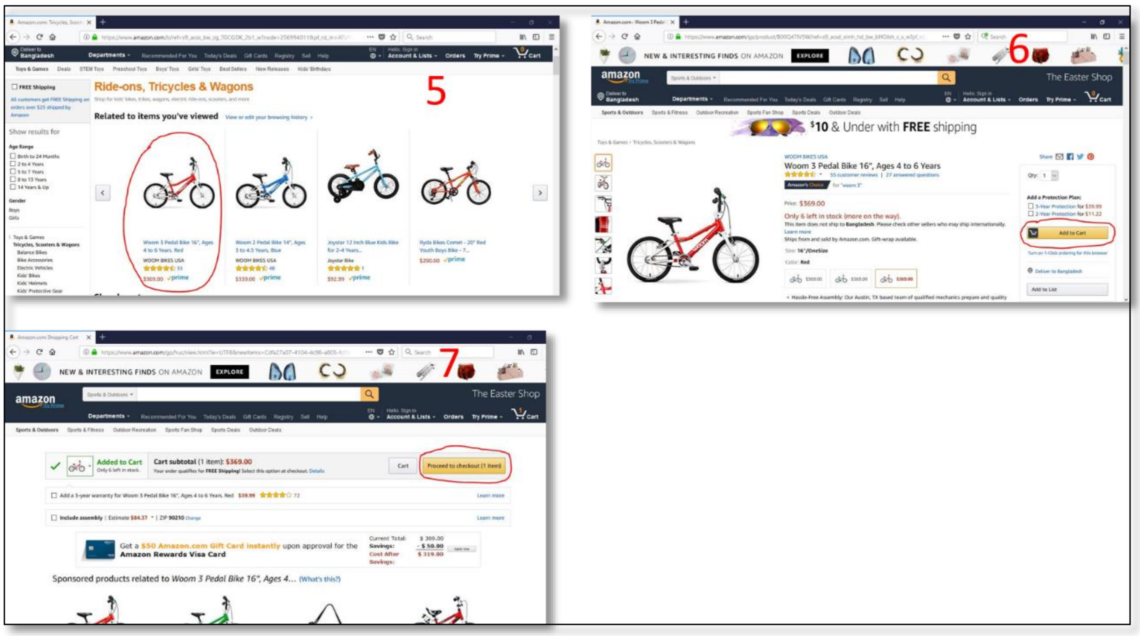


Figure 15: sequence of targets to select for subtasks 6 to 8 in the e-commerce website experiment

Using the mouse, each participant completed the task described above by executing the sequence of subtasks in order until the last step. The time to complete the task was recorded for each participant. Similarly, participant performed the same task using our multimodal input method and the completion time was recorded as well. As mentioned before, we used the Wizard of OZ technique to simulate a unimodal voice-controlled browser. To complete the task using this scenario, an operator executed the mouse click operations following instructions from a participant. The instructions involved reading aloud the text display on a button or hyperlink. After reading out the text completely and clearly, the operator executed the action by clicking on the appropriate button or link at each stage. Participants were informed of this set-up before starting the experiment. Though this was not a real implementation of a voice-controlled web browser, it however revealed the characteristics and approximate time it might take to complete such a task using a voice-controlled web browser. Table 7 shows the task completion time in seconds for each participant for the mouse, voice-only method and the multimodal input method for the e-commerce experiment.

Table 7: Task completion time in milliseconds for the e-commerce website (Amazon.com) case-study

Participant	Task completion time in seconds		
	Mouse	Voice-only	Multimodal Input
1	16.6	30.1	33.0
2	18.8	34.4	33.8
3	15.3	30.8	31.0
4	17.3	38.1	30.0
5	19.5	31.7	29.8
6	17.7	55.9	30.2
7	15.5	38.8	30.5
8	18.1	38.5	36.9
9	18.5	38.5	30.3
10	17.5	37.6	29.9
11	16.6	35.3	30.5
12	18.0	38.7	34.0
Average	17.4	37.4	31.7
SD	1.28	6.69	2.24

The table below compares the KLM estimated task completion time and the actual task completion time (average) for the multimodal input interface. Note that $R(t)$ is the system response time which depends on the internet speed and system used.

Table 8: Comparison of KLM estimated task completion time and actual task completion (average) for the e-commerce task

Estimated time (seconds)	Actual time (average) (seconds)
Fast mode : $(24.27 + R(t))$ Slow mode: $(27.51 + R(t))$	31.7

Figure 13 below is the bar chart representation of the data in table 7. We can observe from the bar chart that the mouse was the fastest with an average task completion time of 17.4s. The mouse was followed by the multimodal interface method with average task completion time of 31.6s. The voice-only method was the slowest with the highest mean completion time of 37.4s. The bar chart also contain errors bars. The error bars show ±1 standard deviation (SD). The voice-only method had the largest standard deviation of 6.69 as can be seen from the size of the error bar. This indicates that most of the task completion time using the voice-only method was more scattered around the mean than for the mouse and the multimodal input interface. The mouse had the least standard deviation. The task completion time using the mouse were close to each other from participant to participant.

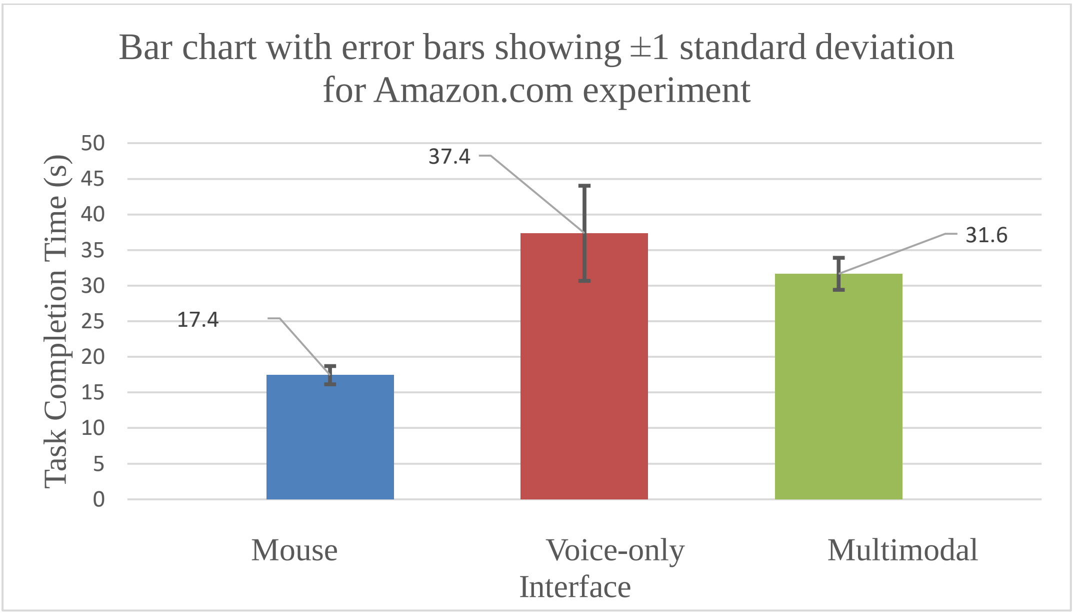


Figure 16: Bar chart with ±1 Standard deviation (SD) error bars for the data in table 5

An analysis of variance (ANOVA) test was performed on the above data to verify the following null hypothesis: *there is no difference in the average time taken to complete the task using the mouse, voice-controlled or multimodal input methods*.

The following results were obtained after doing a one-way ANOVA test on the data in table 6. Note that the independent variable is the navigation method and the dependent variable is the task completion time.

Table 9: ANOVA for e-commerce experiment data set

ANOVA					
Task completion time					
	DF	Sum of Squares	Mean Square	F-value	P-Value
Method	2	2527328172.222	1263664086.111	73.750	.000
Subject	33	565435725.000	17134415.909		
Method * Subject	35	3092763897.222			

The mean task completion time was 17.4s for the mouse, 37.4s for the voice-only input and 31.7s for the multimodal input interface. The mouse was the fastest and the voice-only input was the slowest. The main effect of the interaction method on the task completion time was statistically significant ($F_{2,22} = 73.75, p < 0.05$)

To further determine which interaction method differs from which other interaction method, a Scheffe post hoc comparison test was performed. The table below presents the results of the Scheffe test.

Table 10: Scheffe post hoc comparison test for the e-commerce website experiment

Multiple Comparisons						
Dependent Variable: completion time Scheffe						
(I) method	(J) method	Mean Difference (I-J)	Std. Error	P-Value	95% Confidence Interval	
					Lower Bound	Upper Bound
Mouse	voice	-20.0*	1.7	.000	-24.3	-15.6
	multimodal	-14.3*	1.7	.000	-18.6	-9.9
Voice-only	Mouse	20.0*	1.7	.000	15.6	24.3
	multimodal	5.7*	1.7	.008	1.4	10.0
Multimodal interface	Mouse	14.3*	1.7	.000	9.9	18.6
	voice	-5.7*	1.7	.008	-10.0	-1.4
*. The mean difference is significant at the 0.05 level.						

It can be observed from the table above that the P-Value for each pair (Mouse vs Voice-only, Mouse vs Multimodal interface, and Voice-only vs Multimodal interface) of combination is less than 0.05. Therefore, the mean task completion times are significantly different between each pair of combination of the interaction methods. What this implies is that the observed differences in task completion times is not a chance outcome. If we repeat the experiment, we will likely get similar results with the same pattern.

5.2.2. Experiment II-B: Comparison of task completion time on a typical social networking website. Case-Study: Facebook.com

Today, social networks are among the most common websites visited by people on the internet. The vast majority of internet users spent time on social networks. In this second experiment, we designed a task to be completed on a social networking website, typically Facebook.com. To design the task, we asked a series of questions to a pool of users concerning their browsing habits on Facebook. The main question asked was "how do you spend time on Facebook". The majority of answers was something like "Scrolling down Facebook pages, reading and reacting to posts such as liking pictures and reacting with emoticons and sometimes comments". Based on the answers we received, we designed a simple task that involved opening a Facebook page by activating a link, then once on the page, we were required to scroll down to the bottom of the page where there was one image. Next, we had to open the image in a new browser tab and once on the newly opened tab, we would like the image by activating the like button.

The series of steps to complete this task were as follows:

1) Open Facebook page using the page link
2) Scroll down to the bottom of the page
3) Open the image at the bottom of the page in a new tab
4) Go to the new tab
5) Like the image by activating the "like" button

As in the previous experiment, participants were briefed on the task. Each participant had to complete the task using the three different methods. The time to complete the task was recorded at each trial.

Using the designed KML model of our multimodal input system explained in section 5.1, we estimated the time to complete the task as follows.

Table 11: Subtasks involved in experiment II-B with associated KLM operators for our Multimodal input interface

Serial No.	Sub Task	Operation	t_P (s)
1	Open Facebook page by clicking on hyperlinked page name	MP[50,40]V	0.79
2	Scroll down to bottom of page	2V	0
3	Perform right click and select open in new tab option	MP[348,300]V + P[32,24]V	1.56
4	Display new tab	MP[520,30]V	1.96
5	Click the like button	MP[380,30]V	1.79
		Σt_P	6.1

The estimated task completion time,

$$t_{Execute} = 4t_M + \sum t_P + \ 6t_V + R(t) \tag{26}$$

Where $R(t)$ is the total system response time that depends on the internet speed and the system.

In order to get a better and more accurate estimate, we calculated A and W for the target at each step and calculated P using the Fitts' law prediction equation of our multimodal input interface. In table 9 above, P is denoted as $P[A, W]$, where A is the distance to target and W is the target width. By using our Fitts' law equation, we calculate P for each step as follows:

$$t_P = 333 + 387\ x\ \log_2\left(\frac{A}{W} + 1\right)\ ms \tag{27}$$

Using the fast pronunciation mode (table 4) with $t_V = 0.91s$, the estimated task completion time for this experiment is given by

$$t_{Execute} = 4(1.35) + 6.10 + \ 6(0.91) + R(t) \tag{28}$$

$$t_{Execute} = \left(16.96 + R(t)\right) s \tag{29}$$

In the slow pronunciation mode, $t_V = 1.45s$, the estimated task completion time becomes

$$t_{Execute} = 4(1.35) + 6.10 + \ 6(1.45) + R(t) \tag{30}$$

$$t_{Execute} = \left(20.20 + R(t)\right) s \tag{31}$$

Therefore, the estimated task completion time in the fast pronunciation mode is 16.96s plus system response time. Meanwhile, using the slow pronunciation time for voice commands, the estimated task completion time is 20.20s plus the system response time. It should be noted that the system response time depends on the internet speed and the system used as well as the nature of the input command. The system response time is the time taken by the system to generate a response following an input from the user. Different input will result in different system response time.

Figure 14 below shows the screenshots at each step as taken during the time of running this experiment. The target to select at each step is circled in red. Screenshot 2 contains the subtask 2 and 3 while screenshot 3 has subtask 4 and 5.

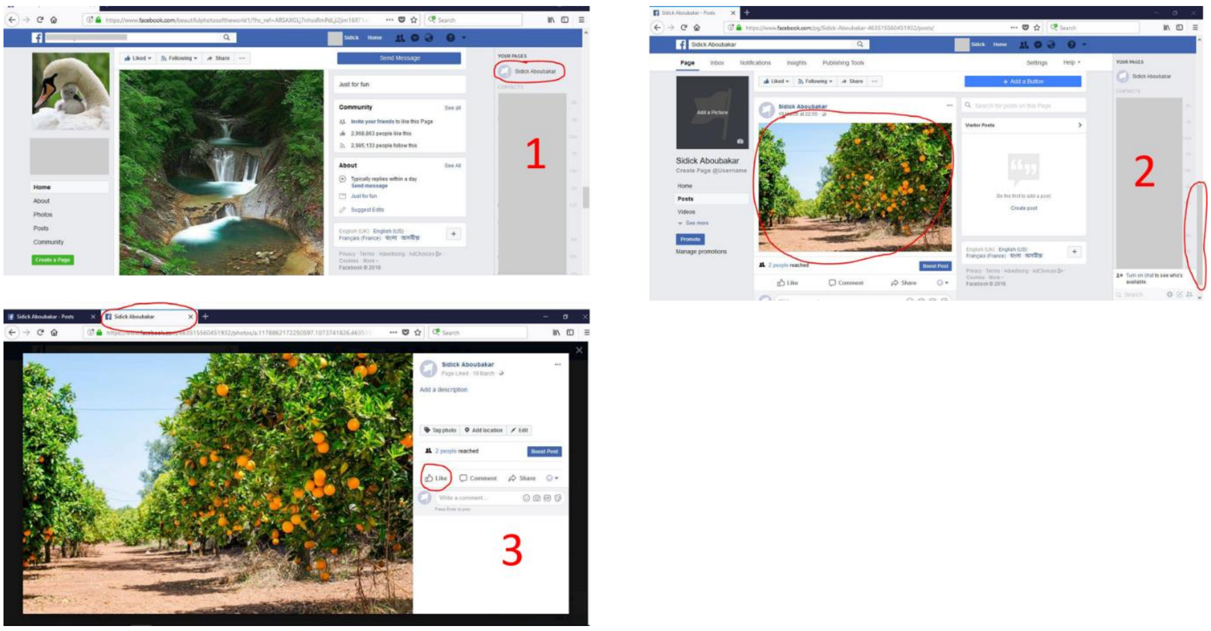


Figure 17: screenshots and targets to select at each subtask on the social network experiment

After going through the sequence of images above, a question might come to mind "why did the user not just use the 'Like' button of the image as it is in step 2?" Or "why open the image again in a new tap since there is already a 'Like' button attached to the image at the bottom of the page?"

The first answer to this question is that allowing the task to end at the bottom of the page by letting the user activate the "Like" button there on the image, introduces a serious problem of ambiguity when using the voice-only input to execute the task. Because there are more than one image on this page with a "Like" button attached to it, by simple uttering "like" during the voice-only test, there is ambiguity on which of the "Like" buttons to select. See Figure 18 below. So to remedy this ambiguity and proceed with the test, we rather opted to open the image in a new tab. It should be noted that there is no problem of ambiguity when the task is accomplished using the mouse or the multimodal input method. As can be seen clearly on Figure 15, there are 3 buttons labelled with the word "Like" on three different objects. Therefore, it will be confusing during the Voice-only navigation to decide on which of the "Like" buttons to activate when the user echoes the voice command "like". Other ambiguous voice commands can be identified on the same figure, such as "comment" and "share".

The second answer is that it also allowed us to test all the voice commands provided by our multimodal input method. Opening the link in a new tab will require the user to use the multimodal input voice command "right click" to activate the context menu. Thus the experiment covered all the commands in our multimodal input method.

Though this inherent ambiguous situation on social networks suggests that Voice-only navigation and interaction will be difficult on such websites, we however proceeded with our experiment by finding a way around this difficulty as explained above. By circumventing this ambiguous scenario, we conducted the experiment to test the performance of the mouse, the voice-only and the multimodal interface method in terms of task completion time.

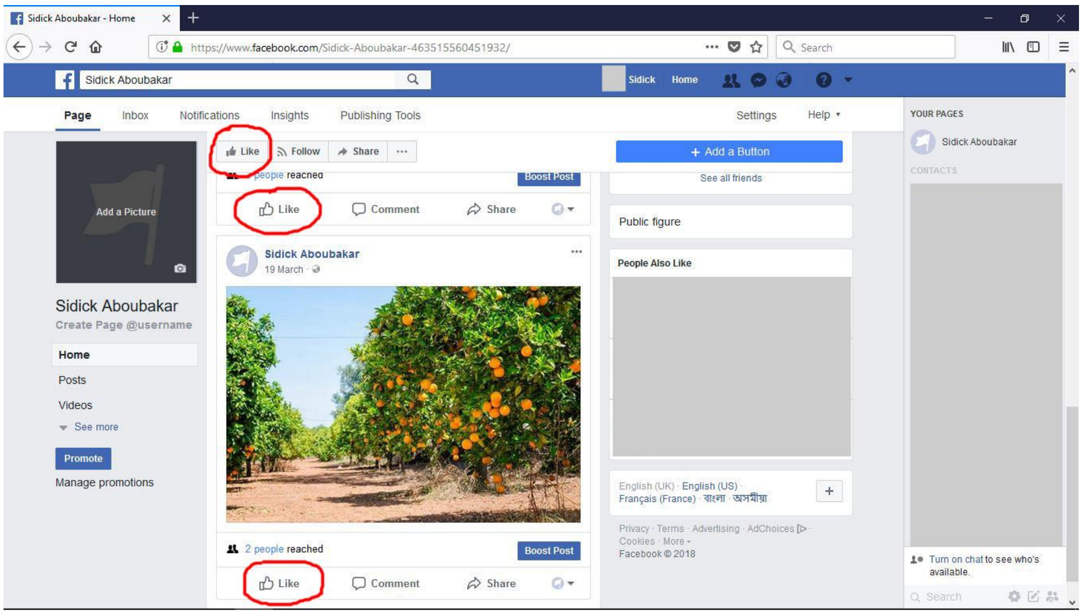


Figure 18: Example of ambiguous scenario for Voice-only interaction on a Facebook page with multiple "Like" buttons

The table below shows the task completion time by participants for the mouse, the voice-only method and the multimodal input interface for the Facebook.com experiment.

Table 12: Task completion time in milliseconds for the social network website (Facebook.com) experiment

Participant	Task completion time in seconds		
	Mouse	Voice-only	Multimodal input
1	8.0	16.0	24.8
2	6.4	11.2	24.4
3	8.3	13.4	22.5
4	8.0	14.4	23.5
5	8.5	14.2	22. 0
6	8.6	11.9	24.6
7	8.8	12.4	24.7
8	7.4	16.4	22.1
9	8.1	16.5	21.0
10	7.9	16.0	20.2
11	8.4	16.2	20.9
12	7.5	16.4	21.3
Average	8.0	14.6	22.7
SD	0.7	2.0	1.7

The table below compares the KLM estimated task completion time and the actual task completion time (average) for the multimodal input interface. Note that $R(t)$ is the system response time which depends on the internet speed and system used.

Table 13: Comparison of KLM estimated task completion time and actual task completion (average) for the social network task

Estimated time (seconds)	Actual time (average) (seconds)
Fast mode : $(16.96 + R(t))$ Slow mode: $(20.20 + R(t))$	22.7

The chart in figure 16 below is a summary of the data in table 11. It shows the mean task completion time for each of the three interaction techniques used in the experiment. It also shows error bars with ±1 standard deviation (SD). It is clear from the chart below that the mouse had the lowest average task completion time of 8.0s and therefore was the fastest, followed by the Voice-only method with 14.6s and our multimodal input interface came last with a mean task completion time of 22.7s.

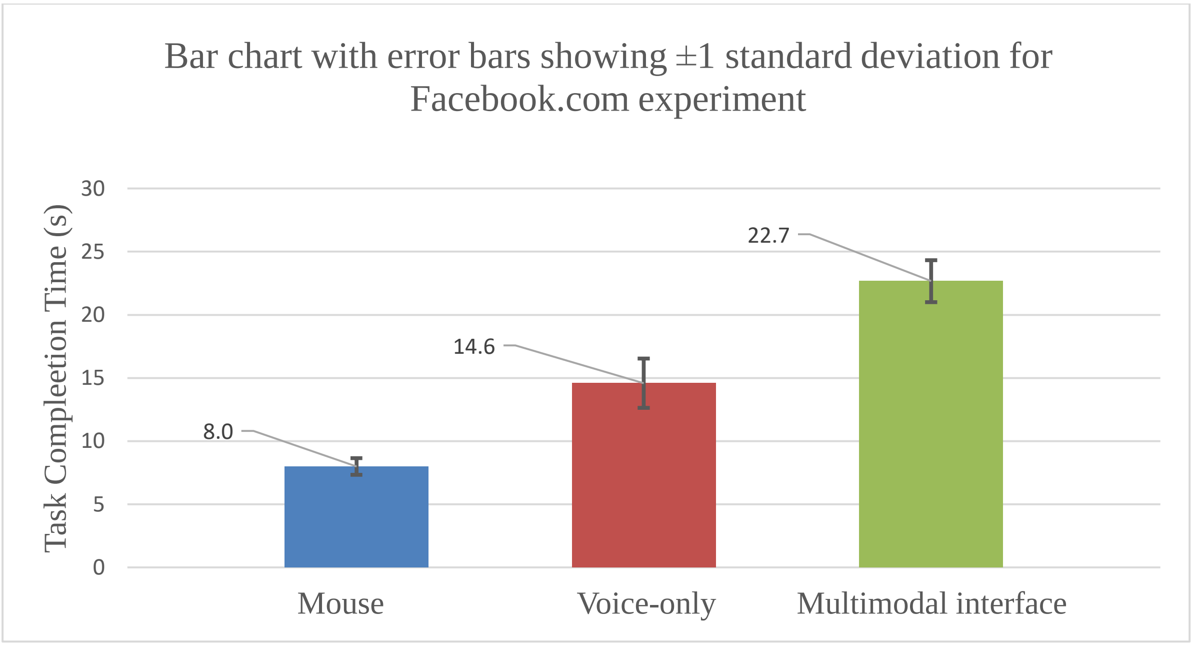


Figure 19: Bar char with ±1Standard deviation (SD) error bars for the data in table 10

As explained before for the e-commerce website experiment, an ANOVA test was also conducted on the data above to determine the significance of the observed difference in task completion times as seen in table 10 above.

The table below shows the results of ANOVA on the data.

Table 14: ANOVA for the social networking website experiment

ANOVA					
Completion time					
	DF	Sum of Squares	Mean Square	F	P-Value
Method	2	1296096955.556	648048477.778	277.409	.000
Subject	33	77090475.000	2336075.000		
Method * Subject	35	1373187430.556			

As can be seen from the table, the result was statistically significant ($F_{2,22} = 277.409, p < 0.05$). However, the multimodal interface was the slowest in this task with mean task completion time of 22.7s. The mouse remained the fastest at 8.0s.

A Scheffe post hoc comparison on the data gave the following results.

Table 15: Scheffe post hoc comparison for the social networking website

Multiple Comparisons						
Dependent Variable: completion time Scheffe						
(I) method	(J) method	Mean Difference (I-J)	Std. Error	P-value	95% Confidence Interval	
					Lower Bound	Upper Bound
Mouse	Voice-only	-6.6*	6.2	.000	-8.1	-4.9
	multimodal	-14.7*	6.2	.000	-1.6	-1.3
Voice-only	Mouse	6.6*	6.2	.000	4.9	8.1
	multimodal	-8.1*	6.2	.000	-9.7	-6.5
Multimodal interface	Mouse	14.7*	6.2	.000	13.1	16.3
	Voice-only	8.1*	6.2	.000	6.5	9.7
*. The mean difference is significant at the 0.05 level.						

Similar to the previous experiment, we observe that the P-value for each pair of combination of the different interaction methods is less than 0.05. Therefore the observed difference in average task completion time is statistically significant between each pair for combination of interaction methods.

5.3 Observation

When we compare the task completion time of the three different methods for the e-commerce website experiment, we see that the mouse performed best with an average time of 17.4s. Our multimodal input method followed with an average time of 31.7s. The voice-only method had the longest task completion time of 37.4s. It should be noted that the time to complete a task using the mouse or multimodal input interface depends on the number of targets to select during the execution of the task. However, in case of the voice-only input, not only does the time depends on the number of targets to select but also on the length and complexity of the text to pronounce at each target. Therefore if the same task is repeated, but this time with the length of the text on the target at each stage increased, the task completion time for the voice-only interaction will likely increase meanwhile the task completion time for the mouse and the multimodal input method will more or less remain similar.

When we look at the average task completion times for the social networking website experiment we notice that the pattern is not the same as with the e-commerce website experiment. Though the mouse was still the fastest with a mean task completion time of 8.0s, this time around, the voice-only interaction was faster than the multimodal input interaction. The voice-only input had an average task completion time of 14.6s which was less than 22.7s for the multimodal input interface. An explanation to this difference is related to the length and complexity of the text on the target at each stage. In this task, the targets have simple words/phrases to be pronounced by the user during the voice-only test. The phrases “Sidick Aboubakar”, “focus”, “open in new tab” and “like” are relatively short and easy to pronounce phrases. The task completion time for the voice-only method will have certainly changed if the length of the phrases where decreased or increased. For example, if the name of the page was changed to “Beautiful Photos of Mountains in Bangladesh”, then the completion time will have increased because it will take more milliseconds to pronounce out “Beautiful Photos of Mountains in Bangladesh” than “Sidick Aboubakar” in order to open the page and continue to the next stage of the task. However, with the page name “Beautiful Photos of Mountains in Bangladesh”, the task completion time for the mouse and the multimodal input method will remain almost the same because these are pointer-based interfaces and the time taken to activate a target using a pointer-based interface is more or less independent of the length and complexity of the text on the target. Therefore, if the same task is repeated under the same system set-up but with the length of the text on each target increased, the average task completion time for the voice-only input will increase, while that of the mouse and multimodal input interface will remain closely the same.

Another problem which was already mentioned before when using voice-only navigation for social networks in particular is the problem of ambiguity in the interaction. A social network like Facebook.com is full of different posts each with attached buttons or links with the same foreground text. Therefore it will be ambiguous to decide on which action to take since there are more than one post or target with links or buttons having the same text. In some studies, they tried to solve this problem by using numbered links. When links are numbered, users will be required to read out the number adjacent to a link in order to activate it. However, this method was found to be difficult to use. The numbering might grow large in case of the social network making it more difficult for users to manage.

One thing we did not address in our experiment is text entry on forms. With the help of a virtual keyboard, characters can be typed using our multimodal input pointing system. The time required to type one character from a virtual keyboard using our system can be estimated using the KLM model we built and described in section 5.1. The task of typing a character from a virtual keyboard can be broken down into subtasks. These are presented below with their associated KLM operators

1) Mentally prepare for action **M**
2) Position mouse pointer over desired character **P**
3) Utter voice command "left click" to activate character **V**

Therefore, the time to enter a single character from an onscreen virtual keyboard is equivalent to

$$t_{Exectute} = t_M + t_p + t_v \qquad (32)$$

Equation (24) is an estimate of the time required to type a character assuming the virtual keyboard is already active. Initially, the user will have to activate the virtual keyboard before beginning character entry. This introduces additional time equivalent to equation 32. The user will mentally prepare to open the virtual keyboard, then move the mouse pointer using head movements to the virtual keyboard icon and finally open the virtual keyboard by echoing the voice command equivalent to a left click. From there, the user can start entering characters following the subtasks listed above. It should be noted that for a word consisting of more than one character, the subtask 1 above (Mentally prepare for action) can be omitted for subsequent characters after the first character is typed. This is because the user is prepared to enter a word which he/she already knows the sequence of characters constituting that word. So less or no time is required to think of what character to type next after the user has begun typing the word.

We received position feedback from the participants who took part in the experiments we conducted. Some participants commented that the voice commands were easy to use and to remember. One participant said he enjoyed using the multimodal input system as a replacement to the mouse. He expressed satisfaction towards using the system. On the other hand, another participant complaint about having difficulties moving the mouse pointer using head movements. The participant said at some point, he felt uncomfortable stretching his head vertically. However, he did not feel any discomfort when he moved his head horizontally. He said it was relatively easy to move the head horizontally than vertically. Other participants complained about the relative stability and accuracy of moving the mouse pointer with our system as compared to when using the standard mouse. Despite this difference in stability and accuracy, they however appreciated the effectiveness and usefulness of the system.

6. Conclusion and Future Works

6.1. Conclusion

To conclude, we have developed and evaluated a multimodal input interface that combines head movements and voice commands to simulate the standard mouse as a pointing system. The multimodal system tracks head movements to control the mouse pointer on the screen and uses voice commands to trigger mouse button actions on the cursor. This multimodal interface is targeted at people with upper limb motor disabilities as an alternative to the standard mouse. We constructed a Fitts' law predictive model for our system to predict the pointing time using head movements. We equally built a KLM model for predicting the time required to complete a task using our multimodal input system. We evaluated the performance of our system in terms of task completion time on web navigation. We compared this performance with that of the standard mouse and the voice-only interaction methods. We found out that the mouse was faster than our system in all cases. However, our system performed better than the voice-only input and our system also addressed some of the challenges of voice-only navigation such as ambiguity, unstructured commands, and long list of command vocabulary.

During the development of the system, we however faced some challenges. Stabilising the movement of the mouse pointer on the screen was one of the major challenge. Initially, there was a lot of jitter associated with the mouse pointer as a user tried to control it using head movements. This instability is the result of the high inertia of head movements. Unlike the hand which can be moved and stopped with good precision, it is not the case with the head. By using the Kalman filter and optimising the size of the virtual mousepad we greatly reduced the jittering and achieved a smooth and stable movement of the mouse pointer.

Another challenge we faced was setting an optimum size for the virtual mousepad. Making the size of the rectangular mousepad too large resulted in stressed movements on the part of the user. A larger length of the virtual mousepad would mean that the user will have to strain when moving his/her head horizontally in either directions. The effects of a large virtual mousepad could even be more noticed with vertical movements. A large width meant users had to stretch their head vertically during controls that required moving the mouse pointer to the top or bottom of the screen. This could easily lead to neck pain. On the other hand, a relatively small virtual mousepad size resulted in rapid displacements of the mouse pointer on the screen with slight head movements. This potentially reduced the precision of motion. Therefore, we had to overcome this challenge by empirically determined the optimum size of the virtual mousepad in relation to the screen size.

Though this is not much of a challenge, we had to ensure that only one face among the faces detected by the camera is the active face that controls the mouse pointer. Though multiple faces can be on a video frame, the foremost face is the controlling face. Therefore the system has no problem if people stand behind the main user while he/she uses the interface.

One difficulty we have not considerably reduced its effects is that of side commands. If a person sitting near the systems utters words that are same to the voice commands, it is likely to be detected by the system if the person spoke out loud enough. This can have an unwanted effect on the task of the main user. Fully overcoming this problem is part of our future work explained in the next section.

One thing we did not do was to test the interface with people representative of the target group of users. Due to some restrictions, we could not reach out to people in the main target group of users. Though we could not have real users with special needs to test the system, the group of users who took part in the experiment provided positive feedback as to the usefulness of the system especially to the target users. During the experiment, users performed the task using the interface without any assistance and in majority of cases with little or no complain or difficulties.

6.2. Future Works

We have plans to make the list of voice commands customisable. We intend to improve the voice component of the interface in such a way that each user will have the freedom to set his/her own customised phrases for the corresponding voice commands. We also plan to extend the list of voice commands by exploring more actions that can be performed with the combination of the mouse and/or keyboard. As part of improving the voice component, we also plan to work on minimising the effects of side commands. As mentioned above, considerably reducing the effects of side commands was one of our challenges. We plan to see how we can tackle this problem in our future works.

In our future work, we intend to design an experiment to test the multimodal interface with the initial target group of users. To actually get a full and complete picture of the usefulness of the interface to people with special needs as well as to reveal some hidden requirements or design considerations, in our next version, we will endeavour to involve some users with upper limb motor disabilities in the test experiments.

We will equally conduct test experiments to determine the performance of the multimodal input interface when operating in the break-reset tracking mode as well as its performance in other application domains.

7. References

[1] Alexander Gruenstein, Ian McGraw, Ibrahim Badr, “The WAMI toolkit for developing, deploying, and evaluating web-accessible multimodal interfaces,” in *Proceedings of the 10th international conference on Multimodal interfaces*, Chania, Crete, 2008.

[2] Kevin Christian, Bill Kules, Ben Shneiderman, Adel Youssef, “A comparison of voice controlled and mouse controlled web browsing,” in *Proceedings of the fourth international ACM conference on Assistive technologies*, Arlington, Virginia, 2000.

[3] Pourang Irani, Sharon Oviatt, Matthew Aylett, Gerald Penn, Shimei Pan, Nikhil Sharma, Frank Rudzicz, Randy Gomez, Ben Cowan, Keisuke Nakamura, “Designing Speech, Acoustic and Multimodal Interactions,” in *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems* , Denver, Colorado,, 2017.

[4] Lee, Vicki L. HansonJohn T. RichardsChin Chin, “Web Access for Older Adults: Voice Browsing?,” in *International Conference on Universal Access in Human-Computer Interaction*, Verlag Berlin Heildelberg, 2007.

[5] S. L. Hura, “Usability Testing of Spoken Conversational Systems,” *Journal of Usability Studies,* vol. 12, no. 4, pp. 155-163, 2017.

[6] Rajeev Agarwal, Yeshwant Muthusamy, Vishu Viswanathan, “Voice Browsing the Web for Information Access,” 1998. [Online]. Available: https://www.w3.org/Voice/1998/Workshop/RajeevAgarwal.html.

[7] Yael Dubinsky, Tiziana Catarci, Stephen Kimani, “A User-Based Method for Speech Interface Development,” in *In: Stephanidis C. (eds) Universal Acess in Human Computer Interaction. Coping with Diversity. UAHCI 2007. Lecture Notes in Computer Science*, Berlin, Springer, 2007, pp. 355-364.

[8] Karl Lewis, Pettey Micheal, Shneiderman Ben, “Speech Activated versus Mouse-Activated Commands for Word Processing Applications: An Empirical Evaluation,” *International Journal of Man-Machine Studies,* pp. 667-687, 1993.

[9] S. Oviatt, “Interface techniques for minimizing disfluent input to spoken language systems,” in *CHI '94 Conference Companion on Human Factors in Computing Systems*, Boston, 1994.

[10] E. Protalinski, “VentureBeat.com,” 17 05 2017. [Online]. Available: https://venturebeat.com/2017/05/17/googles-speech-recognition-technology-now-has-a-4-9-word-error-rate/. [Accessed 11 2017].

[11] Jhilmil Jain, Arnold Lund, Dennis Wixon, “The future of natural user interfaces,” in *CHI EA '11 CHI '11 Extended Abstracts on Human Factors in Computing Systems* , Vancouver, 2011.

[12] Andreia Sias Rodrigues, Vinicius Kruger da Costa, Rafael Cunha Cardoso, Marcio Bender Machado, Marcelo Bender Machado, Tatiana Aires Tavares, “Evaluation of a Head-Tracking Pointing Device for Users with Motor Disabilities,” in *Proceedings of the 10th International*

Conference on PErvasive Technologies Related to Assistive Environments, Island of Rhodes, 2017.

[13] Patle Pooja, Waigaonkar Snehal, Patil Jyoti, Anjankar Piyush, “A Camera Mouse- Applicaiton for Disable Person: A review,” *International Journal of Engineering Science and Computing,* vol. 7, no. 9, 2017.

[14] João M.S.Martins, João M.F.Rodrigues, Jaime A.C.Martins, “Low-cost natural interface based on head movements,” in *International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Infoexclusion (DSAI 2015)*, 2015.

[15] Matthew R. Williams, Robert F. Kirsch, “Evaluation of Head Orientation and Neck Muscle EMG Signals as Command Inputs to a Human-Computer Interface for Individuals With High Tetraplegia,” *IEEE Transaction on Neural Systems and Rehabilitation Engineering,* vol. 16, no. 5, 2008.

[16] Amer Al-rahayfeh, Miad Faezipour, “Eye Tracking and Head Movement Detection: A State-of-Art Survey,” *IEEE journal of translationall engineering in health and medicine,* vol. 1, 2013.

[17] P.C.Anjankar, S.A.Waigaonkar, P.D.Patle, J.D.Patil, “A Camera Mouse - An Application for Disable Person,” *International Journal of Computer Sciences and Engineering,* vol. 6, no. 3, pp. 133-137, 2018.

[18] Tunhua Wu, Ping Wang, Yezhi Lin, “A Robust Noninvasive Eye Control Approach For Disabled People Based on Kinect 2.0 Sensor,” *IEEE Sensors Letters,* vol. 2, no. 3, 2017.

[19] I. Scott MacKenzie, Abigail Sellen, William A. S. Buxton, “A comparison of input devices in element pointing and dragging tasks,” in *SIGCHI Conference on Human Factors in Computing Systems*, New Orleans, 1991.

[20] I. S. Mackenzie, Human-Computer Interaction An Empirical Research Perspective, Elsevier, 2013.

[21] Behnaz Yousefi, Xueliang Huo, Maysam Ghovanloo, “Using Fitts’s Law for Evaluating Tongue Drive System as a Pointing Device for Computer Access,” in *Annual International Conference of the IEEE in Medicine and Biology Society*, 2010.

[22] Jing Kong, Xiangshi Ren, “Calculation of Effective Target Width and Its Effects on Pointing Tasks,” *IPSJ Journal,* vol. 45, no. 5, pp. 1570-1572, 2006.

[23] João M.S.Martins, João M.F.Rodrigues, Jaime A.C.Martins, “Low-cost Natural Interface Based on Head Movements,” in *Proceedings of the 6th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion*, 2015.