PINPOINTING USER INTERFACE DEFICIENCIES

USING PATTERN RECOGNITION

TECHNIQUES

THESIS

Presented to the Graduate Council of
Texas State University-San Marcos
in Partial Fulfillment
of the Requirements

for the Degree

Master of SCIENCE

by

Dasari Kali Venkata Divya, B.E.

San Marcos, TX
May 2013

PINPOINTING USER INTERFACE DEFICIENCIES

USING PATTERN RECOGNITION

TECHNIQUES

Committee Members Approved:

_____
Dan Tamir, Chair

_____
Komogortsev Oleg

_____
Gao Byron

Approved:

_____
J. Michael Willoughby
Dean of the Graduate College

ACKNOWLEDGEMENTS

I would like to thank my family for allowing me the time to pursue this research. In addition, I would also like to thank Dr. Oleg Komogortsev and Dr. Byron Gao for serving on my committee. Finally, I would like to thank Dr. Dan E. Tamir for all his invaluable guidance and assistance during this process.

This manuscript was submitted on December 21, 2012.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

ABSTRACT

PINPOINTING USER INTERFACE DEFICIENCIES

USING PATTERN RECOGNITION

TECHNIQUES

by

Dasari Kali Venkata Divya

Texas State University-San Marcos

May 2013

SUPERVISING PROFESSOR: Dan Tamir

The *Effort Based Model* of usability aids in evaluating user interface (UI), development of usable software, and pinpointing software usability defects. In this context, the term pinpoint analysis refers to identifying and locating software usability issues and correlating these issues with the UI software code. In this thesis, the underlying theories of the effort based model along with pattern recognition techniques are used to produce a framework for identifying usability deficiencies in software.

Often, when users are in a state of confusion and not sure how to proceed using the software, they tend to gaze around the screen trying to find the best way to complete a task. This behavior is referred to as excessive effort. In this work, pattern recognition techniques are applied to data gathered throughout user interaction with software in an attempt to identify excessive effort segments. This is done by logging all user activities as video and data files by an eye tracker. The data files are divided into segments using event based segmentation, where a segment is the time between two consecutive keyboard/mouse clicks. Subsequently, data reduction programs are run on the segments for generating feature vectors. Pattern recognition techniques like feature selection, thresholding, clustering, and principal component analysis (PCA) are applied to the features in order to automatically classify each segment into excessive and non-excessive effort segments. This allows developers to harness their effort and focus on the excessive effort segments that need attention.

To verify the results of the pattern recognition procedures, the video file is manually classified into excessive and non-excessive segments and the results of automatic and manual classification are compared. Experiment results show more than 40% reduction in time for usability testing. Of all the methods used, experiments using the threshold method using the number of fixations and a threshold method applied to the first principal component produce good results which are significantly better than results obtained through other experiments.

CHAPTER 1

INTRODUCTION

One of the primary goals of software is to simplify various tasks and enable users to accomplish tasks with ease and efficiency. Due to the importance of software, numerous fields have recently witnessed an increase in development and deployment. Nevertheless, feedback from software applications end-users consistently shows that software is at times confusing, counter-productive, and unsatisfactory. Clearly, if the user experiences problems or difficulty, it is highly unlikely that he/she uses that software again. Hence, it is very important for software engineers to put emphasis on testing in order to eliminate user complaints and provide the user with good experience.

Software engineers use a wide variety of tools such as prototyping, inspection, usability testing, iterative processes, etc. [1-5], to ensure that the software they produce is usable. Still, these tools may not address the usability problem efficiently, resulting in a low ranking on usability for several systems [6]. The classical methods used in identifying usability techniques have not proven to be very proficient in accurately pointing the specific segment of code that could be leading to the usability problems. Without proper data to understand which part of code is faulty, developers have had a hard time identifying and fixing code that lead to usability issues.

The usability testing process involves observing users engaged with a software application and obtaining a set of characteristics of the user experience. This

methodology requires an expert to construct, conduct, and assess the tests; devoted laboratory facilities to obtain good results; and several users that participate in the tests. Despite all these efforts, usability testing only indicates that a problem exists but does not identify the cause for the problem [7]. This makes usability testing expensive, time consuming and frustrating for both developers and managers and hence, it is often ignored.

Most of the tools used for evaluating usability of a software application use 'time to complete a task', referred to as time on task (ToT), as a measure for evaluating usability [2, 8-9]. This approach of giving high weight to time on task may not produce accurate results when factors like system, network performance or interface design, which are difficult to avoid, play a role. An alternate approach is to measure usability in terms of user-effort, which eliminates some of the system issues mentioned earlier, is allowing software engineers to put more focus on the interface design [7].

The *Effort based model* of usability aids in evaluating user interface, development of usable software, and pinpointing software usability defects [10]. The model is an efficient and relatively inexpensive means for evaluation of software usability. It is developed using the principle that usability is an inverse function of effort. A metric of software usability that is deduced by using the effort based model is used for comparison of different implementations of the same application. The results of several experiments conducted on the effort based model show strong relationship between effort and usability; and also reveal that understandability, operability, and learnability of software systems is measured using the effort based metric [10]. A detailed explanation is provided in section 2.3.

Problem Definition

The underlying theory of the Effort Based Model is used to produce a framework to identify usability deficiencies in the software. Identifying and locating software usability issues and correlating these issues with UI software code is referred to as Pinpoint Analysis. For example, users who are in a state of confusion, and users that are not sure how to use the software, tend to look around the screen to figure out the best way to accomplish a task. This behavior is referred to as an excessive effort. Identifying and pinpointing excessive effort behavior helps UI designers rectify numerous usability related issues. This research attempts to evaluate the utility of pinpointing user interface deficiencies using pattern recognition techniques for identifying excessive effort in segments of software interaction session records. Segmentation of user's software interaction session can be done using the time slice between two consecutive mouse/keyboard clicks. Automatic identification of segments showing excessive effort behavior helps the UI designers to reduce the time required for analysis and rearranging the interface at the pinpointed time snapshot. Some pattern recognition methods used in this work are feature selection, principal component analysis, K-means clustering, and threshold based classification [12].

Hypothesis

The hypothesis of this thesis is that it is feasible to devise a framework that can identify excessive effort segments by applying pattern recognition techniques such as K-means clustering algorithm, thresholding, principal component analysis, and feature selection.

Methodology s

Several experiments are conducted to validate this hypothesis. Developers constantly strive to develop and improve software so as to reduce the effort and make the software experience for a user intuitive. However, there are several places in execution where the developed software has usability issues. With this assumption, this work aims to efficiently identify the specific pieces of code that lead to usability issues. Pattern recognition techniques and effort based model of usability are used to achieve this.

Contribution

In this work I have developed a new methodology using pattern recognition techniques for assessing the usability of software. This methodology helps optimize the time spent on usability testing while also more accurately identifying specific segments of code that could be leading to the usability issues.

CHAPTER 2

BACKGROUND

2.1 Software Usability

According to the International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) 9126 standard, software usability is: “The capability of a software product to be understood, learned, used, and be attractive to the user when used under specified conditions.” There are several characteristics that play an important part in defining software usability: understandability, learnability, operability, and attractiveness [9, 12].

Understandability helps determine how easy it is to comprehend and use the software. It is the ability of a user to understand the capabilities of the software and its suitability to accomplish specific goals. Learnability indicates the ease with which a user learns to use specific software. Operability is the capability of a user to use the software to accomplish a specific goal. The end-goal of any software is to perform a task efficiently. As such, operability plays an important role in usability. Attractiveness relates to the requirement that the end-user experience is pleasant and rewarding.

2.2 Classical Methods for Measuring Usability

This section lists various classical methods that are currently used to evaluate usability. The classical methods are broadly classified into methods that make use of data

gathered from users and methods that rely on usability experts. There are usability evaluation methods that apply to all stages of design and development, from product definition to final design modifications. Usability methods are further classified into *cognitive modeling methods*, *inspection methods*, *inquiry methods*, *prototyping methods*, and *testing methods*.

Cognitive modeling involves creating a computational model to estimate how long it takes the users to perform a given task. It involves one or more evaluators inspecting a user interface by going through a set of tasks by which understandability and ease of learning are evaluated. The user interface is often presented in the form of a paper mock-up or a working prototype; but, it might be a fully developed interface. Cognitive models are based on psychological principles and experimental studies to determine times for cognitive processing and motor movements. They are used to improve user interfaces or predict problem areas during the design process.

The inspection method involves cognition with emphasis on a hands-on approach. Under the inspection methods, the experimenters are used to observe users while they use the software. The testing and evaluation of programs is done by an expert reviewer. This provides quantitative data; as tasks can be timed and recorded. In addition to quantitative data, qualitative user experience data are also collected. Although the data collected are subjective, they provide valuable user critique information.

Experts obtain information about users' likes, dislikes, needs, and understanding of the system by talking to them, observing them using the system, or letting them answer questions verbally or in written form. Since this information is collected by inquiring and getting direct feedback from users, this model is called the inquiry method. While the

above methods focus on usability testing at an advanced stage in the development, the prototyping method tries to improve usability by refining and providing feedback as the software is being developed. Rapid prototyping is a method used in early stages of development to validate and refine the usability of a system. It is used to quickly and efficiently evaluate user-interface designs without the need for an expensive working model. This helps removing the developer's resistance to design changes since it is implemented before any actual programming begins.

Testing methods provide usability evaluation through testing of users for the most quantitative data. User interaction sessions are recorded on video that provides task completion time and allows for observation of user attitudes.

2.3 The Effort Based Usability Model

Several studies indicate that many system users associate the "physical" effort required for accomplishing tasks with the usability of the software [10, 13]. The effort based model for software usability stems from the notion that usability is an inverse function of effort. For example, an eye tracking device is used to measure the effort expanded by the user in navigating through the user interface of software. Physical and mental effort are obtained and inferred from logging user activity. For this model, *E* denotes the total effort required to complete a task with computer software and is defined as:

$$E = \begin{pmatrix} E_{mental} \\ E_{physical} \end{pmatrix}$$

$$E_{mental} = \begin{pmatrix} E_{eye\_mental} \\ E_{other\_mental} \end{pmatrix}$$

$$E_{physical} = \begin{pmatrix} E_{manual\_physical} \\ E_{eye\_physical} \\ E_{other\_physical} \end{pmatrix}$$

Where:

E_{eye_mental} denotes the amount of mental effort to complete the task measured by eye related metrics.

E_{other_mental} denotes the amount of mental effort measured by other metrics.

$E_{physical}$ denotes the amount of physical effort needed to complete the task.

$E_{manual\_physical}$ denotes the amount of manual effort required to complete the task. Manual effort includes, but is not limited to, the movement of fingers, hands, arms, etc.

$E_{eye\_physical}$ denotes the amount of physical effort measured by eye movement related metrics.

$E_{other\_physical}$ denotes the amount of physical effort measured by other metrics.

2.3.1 Measuring Effort

As stated above, the effort required to complete tasks is associated with software usability [6]. Physical effort includes manual effort and physical eye effort. In the case of

interactive computer tasks, it is possible to calculate effort as a linear combination or a weighted sum of metrics such as the number of mouse clicks, number of keyboard clicks, total eye path traversed, total mouse path traversed.

Mental effort is essentially the amount of brain activity required to complete a task. To some extent, brain activity related to a task is approximated by processing eye movement data recorded by an eye tracker [1, 4, 10, 14-16]. Eye trackers acquire eye position data and enable classifying the data into several eye movement types useful for eye related effort assessment. The main types of eye movements are: 1) *fixation* – eye movement that keeps an eye gaze stable with regard to a stationary target providing visual pictures with high acuity, 2) *saccade* –rapid eye movement from one fixation point to another, and 3) *pursui***t** – stabilizes the retina with regard to a moving object of interest [10]. Usually, the Human Visual System (HVS) does not exhibit pursuits when dynamically moving targets are not a part of the interface [10].

Many researchers consider the following metrics as a measure of the *physical* and/or *cognitive load* [15]. Hence, these metrics facilitate the estimation of mental effort.

- **Average fixation duration:** Average fixation duration, generally measured in milliseconds, indicates cognitive load that is interpreted as a difficulty in extracting information or as an indication that an interface object is engaging [14, 15].
- **Average saccade amplitude:** Saccade amplitude, measured in degrees, indicates meaningful cognitive load cues. Large average saccade amplitude indicates high mental effort.
- **Number of saccades:** High number of saccades indicates increased effort to accomplish a task [14].

- **Number of fixations:** High number of fixations indicates increased effort to accomplish a task [14].
- **Average eye path traversed:** Average eye path traversed is the average of the distance traversed by the eyes between two consecutive fixation points during a task. The length of the path traversed by the eye is proportional to the effort expended by the HVS to achieve the goal.

2.3.2 Steps in conducting an effort based usability evaluation

The effort based usability evaluation is broadly divided into three phases: *Measurement, Analysis, and Assessment* [2, 13].

In the *measurement process*, a group of users executes a set of identical independent tasks, which emerge from a single scenario. These tasks differ in key parameters which prevent the users from memorizing a sequence of interaction activities. Throughout the interaction process, certain user activities like eye movement, time on task, keyboard, and mouse activities are logged.

The *analysis* phase involves accumulating data for several metrics such as the number of saccades, average saccade amplitude, number of fixations, average fixation duration, average eye path traversed, etc., that relate to user effort. One other metric is the time on task. The average task completion time is compared to a learning curve which reflects users' mastery of software.

The final step is the *assessment*. Using the above steps, the learnability of software systems is assessed and the point of users' mastery of software is identified. The same model is applied to obtain operability and understandability of various systems or

different groups of users using the same system. The effort based metric measurement provides interface designers and developers with a methodology to evaluate their designs [10].

2.4 Learnability based Usability Model

Typically, as users become familiar with an application, the time to complete tasks which emerge from the same scenario become shorter and shorter. Often, a graph of Effort-On-Task (EoT) averages or Time-On-Task (ToT) for the users, gives an exponential decay curve (E_{avg}) that represents average effort on task expended by the group of users. Figure 1 depicts a typical graph, The E_{exp} line is the effort that the interface designer expects from an expert to expend in order to complete the task. The point where the user's effort reaches the acceptable level is the learning point L_p. The learning time (L_T) is calculated by adding the average task duration to the left of the learning point. Data to the right of the learning point gives the amount of effort required by a trained user to complete tasks. This model is further used to compare two systems and to identify outlier tasks which are studied to find the shortfalls [10].

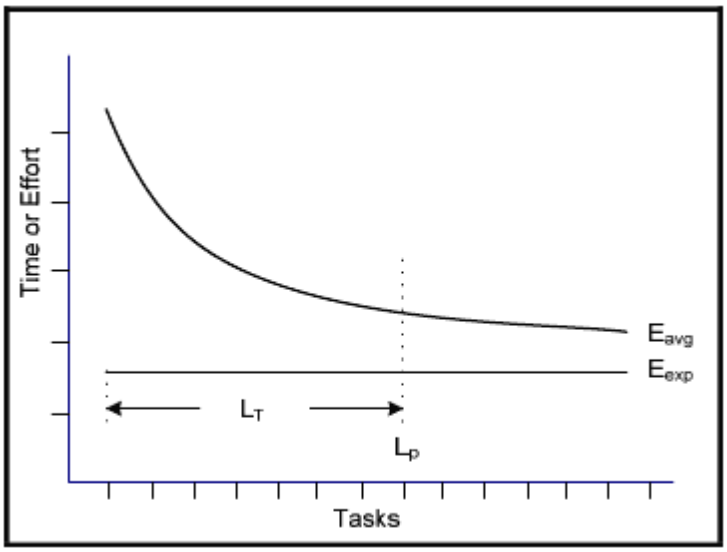

Figure 1 - Learnability based Usability Model

2.5 Pinpointing Usability Issues

Usability testing is considered one of the more expensive, tedious, and least rewarding tests to implement. This perception is likely to change if usability testing is made less expensive and more rewarding. This requires accurate means through which an engineer identifies and pinpoints issues in the software or the interface. This process is called pinpoint analysis. Pinpoint analysis is one of two types; inter-pinpoint analysis deals with identifying issues with tasks performed by the users in a specific system, whereas intra-pinpoint analysis refers to identifying issues within tasks in a specific system. For example, outlier tasks might be identified through inter-pinpoint analysis and used for intra pinpoint analysis. This analysis also helps graphical user interface (GUI) designers to make decisions about element placement on displays and determine the level of effort that is related to different widgets [17].

2.5.1 Inter Pinpoint Analysis

Inter pinpoint analysis involves detecting tasks that present anomalies and identifying the reasons for these anomalies at a high level. The mouse is used as an example to understand inter-pinpoint analysis. In a particular task, the right mouse button helps users complete a task effectively; however, some of the users are unaware of it. It is possible that anomalies like this can be identified in inter pinpoint analysis [17].

Inter pinpoint analysis helps identifying alternative methods to perform a task effectively with less effort; however, it does not provide users with a hint of the alternative method. Other issues like the necessity of help facilities in software are identified by the high level analysis of tasks that present anomalies. Figure 2 is a plot of the average time on task of five subjects for seven identical but independent tasks. The X axis shows a data point for each of the seven tasks, while the Y axis shows information related to time on each task. The curve fitted to the individual bars representing the average time on task is a power law curve that actually corresponds to the learnability model. In this case, task three that does not fit well in the curve shows an anomalistic behavior and calls for further analysis and study [17].

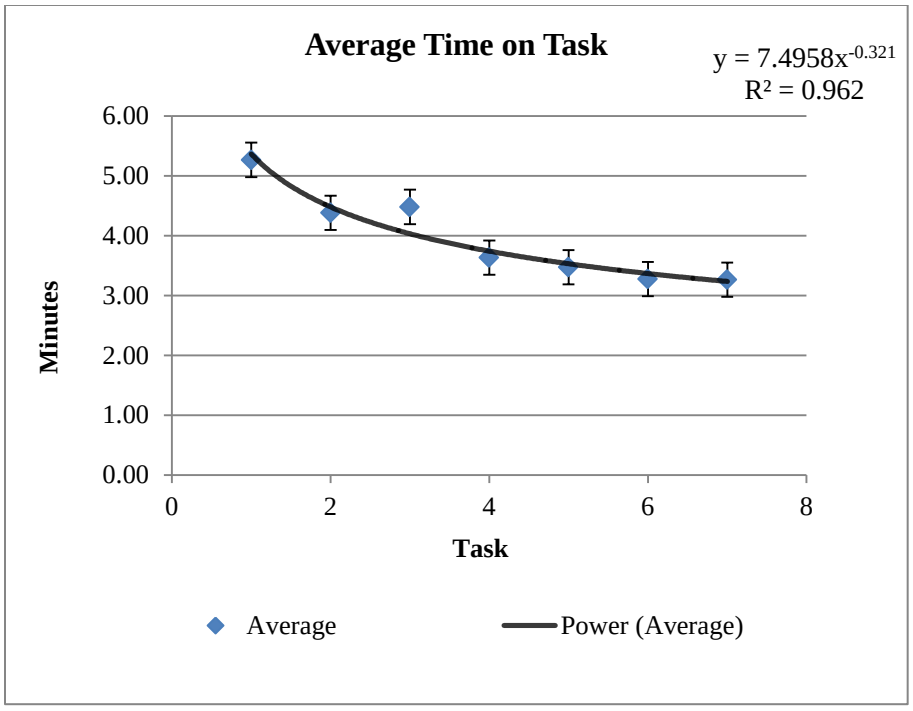

Figure 2 - Inter pinpoint Analysis

2.5.2 Intra Pinpoint Analysis

A more detailed method for analyzing tasks and identifying specific issues with the software is intra pinpoint analysis. Intra pinpoint analysis can be done manually by watching all the video recordings of the users' interactions with software, obtained from an eye tracking device. This review helps identifying interaction issues and areas where the user has difficulty while performing tasks. For example, the analysis might reveal that most of the users go into a state of confusion in a specific part of a task and are looking around the screen to identify the best way to proceed with the task. This might prompt the designers to rearrange the interface at the relevant time snapshot. Clearly, this option is tedious and potentially expensive. An alternative is to use a semi-automatic method applying pattern recognition technique. This method eliminates the need for a person to

watch the entire video in order to identify interaction issues thereby cutting down the cost and time. It enables automatic identification, for further evaluation, of areas where the user has difficulty. A brief description of the pattern recognition techniques that were used in this thesis is discussed in the next section.

2.6 Pattern Recognition

One of the applications of pattern recognition is the assignment of *label*s to a given input value, or *instance*, according to a specific algorithm. An example of pattern recognition is classification, which attempts to assign each input value to one of a given set of classes. Pattern recognition is generally categorized according to the type of learning procedure used to generate the output value. *Supervised learning* assumes that a set of training data (the training set), consisting of a set of instances that have been properly labeled by hand with the correct output, has been provided. Next, a learning procedure generates a model that attempts to meet two sometimes conflicting objectives: Perform as well as possible on the training data, and generalize as well as possible to new data. On the other hand, *unsupervised learning* assumes the availability of training data that has not been hand-labeled and attempts to find inherent patterns that are used to determine the correct classification value for new data instances [18].

2.6.1 Classification

Algorithms for pattern recognition depend on several parameters, such as the type of output labels, and on the training / learning method which are supervised or unsupervised. Additionally, the algorithms differ in the way that inference is performed. For example, inference might be based on probability, on non-parametric clustering,

fuzzy logic, etc. [19, 20]. The following are various relevant pattern recognition techniques.

2.6.1.1 Segmentation

Pattern recognition techniques require the definition of patterns. In this thesis segments of user activities records serve as the basic patterns. In this research, a segment is defined as the time between two consecutive keyboard/mouse clicks.

2.6.1.2 Feature Extraction and Feature Selection

Generally, the objects that are subject to classification, i.e. the patterns (segments in the case of this research), are represented through a set of measurements (say n measurements) or characteristics referred to as features. Hence, the objects are considered as vectors in an n-dimensional space referred to as the feature space. Feature selection also known as variable selection, variable subset selection, feature reduction, and attribute selection, is a technique for selecting a subset of relevant features for building robust learning and inference models [19]. Feature selection algorithms attempt to reduce the dimensionality of the feature space and reduce the complexity of the recognition process by pruning out redundant, correlated, and irrelevant features. There are several feature selection algorithms, some of which are discussed below.

2.6.1.2.1 Exhaustive Search

Exhaustive search is a brute-force feature selection method where all possible subsets of the features are exhaustively evaluated and the best subset is selected. The number of combinations of *r* objects from a set of *n* features is $(n!/(r!(n-1)!)$. This results in a very large set of combinations of features to examine. Hence, generally the

exhaustive search's computational cost is prohibitively high. Thus, this method is impractical if the number of features in the set is large [11]. Because of the problems associated with exhaustive search, people resort to adopting greedier, heuristic, selection algorithms to further enhance the efficiency. In this thesis there are five features of interest. Nevertheless, evaluating all the possible subsets of the five features is time consuming. Hence, due to the complexity of the evaluation process, exhaustive search is not a viable option. For these reasons, the heuristic approach is adapted.

2.6.1.2.2 Heuristic/Suboptimal Search

Heuristic search refers to selecting a feature subset by making an educated guess and finding out if the selection yields good results. Otherwise, the heuristic procedure examines other subsets. It is a good alternative where an exhaustive search is impractical [20].

2.6.1.3 Principal Component Analysis (PCA)

PCA is an unsupervised regression procedure that analyses sample data, such as the set of training patterns, in order to identify a coordinate transformation that de-correlates the data and "orders" the information (or variance) associated with the data in the axes of the new space in a monotonically decreasing fashion. In general, as a result of the transformation, most of the information associated with the data is concentrated in the first few components of the new space. This enables ignoring components (axes) that do not carry significant information, thereby reducing the dimensionality of the space used for pattern representation and recognition. Each principal component is a linear combination of the original variables. The principal components as a whole form an orthogonal basis for the data space [11].

The distinction between the principal component analysis and feature selection is that, following the PCA, the resulting features are different than the original features; they do not correspond directly to the set of measurements, and are not easily interpretable, while the features left after feature selection are simply a subset of the original features.

Following the feature selection and/or PCA, classification is applied via different methods including thresholding, discriminate analysis, decision functions, and clustering [11, 19, 20].

In this research, heuristic based feature selection techniques as well as principal component analysis are used to reduce the dimensionality of the data set consisting of a large number of interrelated variables like the saccade count and the average saccade amplitude, fixation count and average fixation duration, etc., while retaining as much as possible of the variation present in the data set. In the case of PCA, this is achieved by transforming the data set into principal components, which are ordered so that the first few retain the most of the variation present in all of the original variables. The principal components are then subjected to clustering algorithms to find segments of excessive effort.

2.6.1.4 The Threshold Method

Another pattern recognition method is the threshold method that aims to classify input data based on a threshold value. In this thesis, the threshold value is calculated by counting the number of fixations, saccades, key strokes, mouse clicks in each segment and using the means of these values as thresholds. All values greater than the threshold are put into one group while input values below the threshold are classified into a second

group. One problem with the threshold method is that it is limited to one dimensional data. Hence it is only applied to individual features, or a combination of features, such as linear combinations or specific components of the PCA. Clustering techniques, however, are used to efficiently classify multidimensional data.

2.6.1.5 Clustering

Clustering is a multi-disciplinary, widely-used, unsupervised algorithm method to classify data. It involves the assignment of a set of patterns into subsets (called clusters) so that patterns in the same cluster are similar in some sense. To define a cluster, it is necessary to first define a measure of similarity which establishes a rule for assigning patterns to the domain of a particular cluster center. Generally, and in this thesis, Euclidian distance is used as the distance measure. In Cartesian coordinates, if $p = p_1, p_2, \ldots, p_n$ and $q = q_1, q_2, \ldots, q_n$ are two points in *n* dimensional space, the Euclidean distance between *p* and *q* is:

$$D(p,q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

The Euclidean distance is used as a measure of similarity, the smaller the distance the greater the similarity. There are several clustering algorithms such as the hierarchical, partitional, density based, and subspace clustering algorithms. In this research, however, partitional algorithms are of interest. Partitional clustering involves partitioning of n observations (patterns) into k clusters where each observation belongs to the nearest cluster. The K-means algorithm, used in this research, is a partitional algorithm that attempts to minimize the mean square distance between patterns and cluster centers. The

center is the centroid of all the cluster patterns. The algorithm consists of the following steps [11]:

Step 1: Choose K initial cluster centers $z_1(1), z_2(1), z_3(1), \ldots, z_k(1)$. These are arbitrary and selected as the first K samples. The cluster centers at k^{th} iteration is denoted by $z_i(k)$, where, $i = 1, 2, \ldots, K$.

Step 2: At the k^{th} iterative step distribute the samples $\{x\}$ among the K cluster domains, using the following equation

$$x \in S_j(k) \quad if \quad \| x - z_j(k) \| < \| x - z_i(k) \|$$

for all $i = 1, 2, \ldots, K$, $i \neq j$, where $S_j(k)$ denotes the set of samples whose cluster center is $z_j(k)$

Step 3: From the results of step 2 compute the new cluster centers, such that the sum of the squared distances from all points in $S_j(k)$ to the new cluster center is minimized. The new cluster center is given by

$$z_k(k+1) = 1/N_j \sum_{x \in S_j(k)} x_i, \quad j = 1, 2, \ldots, K$$

where, N_j is the number of samples in $S_j(k)$.

Step 4: Repeat step 2 and step 3 until there is no significant change in the cluster centers, i.e.

if $z_j(k+1) = z_j(k)$ for $j = 1, 2, \ldots, K$, which means the algorithm has converged and the procedure is terminated. Figure 3 demonstrates how the algorithm works.

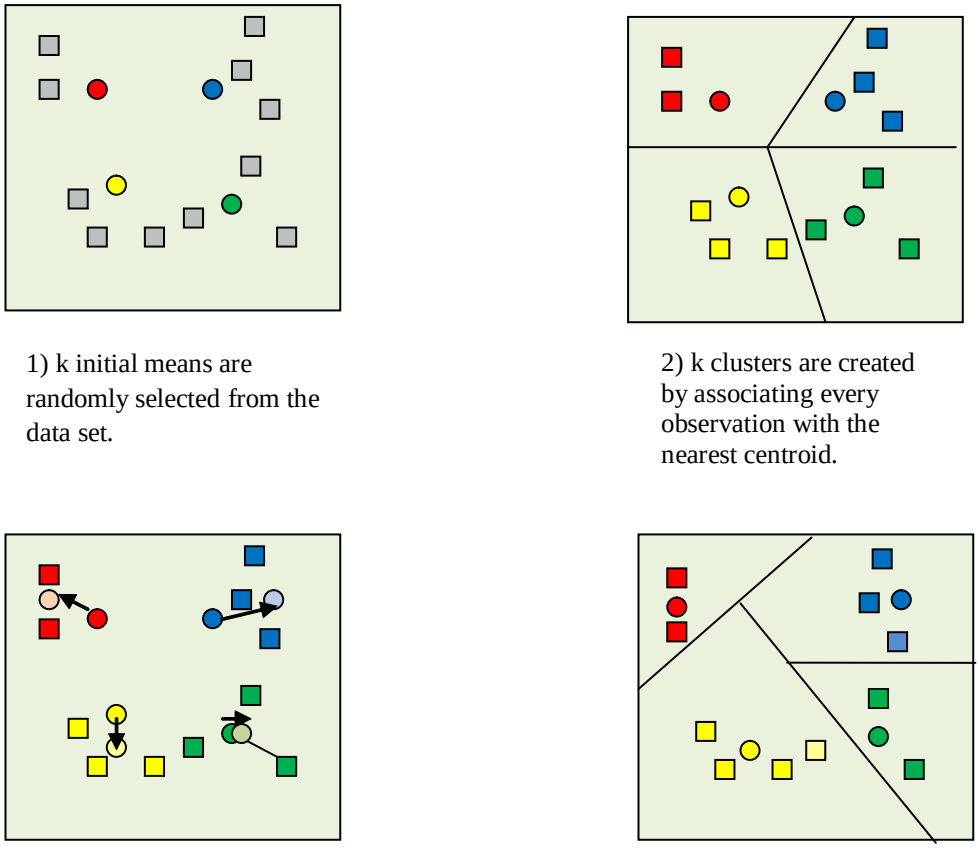

Figure 3 - Demonstration of K-means Algorithm

The advantage of the clustering technique is its ability to classify excessive effort segments by considering a number of features such as saccade count, saccade amplitude, fixation duration, average eye path traversed. In addition, the K-means clustering is used to identify thresholds. MATLAB, a high level language and interactive environment for numerical computation, visualization, and programming [24], has a built in function for

K-means that does exactly as described in the algorithm. This eliminates the need to implement the algorithm.

2.7 MATLAB Functions

MATLAB has several built-in functions that are readily available for use. In this thesis, the MATLAB functions “kmeans” and “princomp” have been used for clustering and principal component analysis.

CHAPTER 3

LITERATURE SURVEY

Usability is a highly researched topic with much literature available. However, extensive searching did not reveal any research papers related to pinpointing usability issues. There are some papers on effort based usability evaluation that are discussed below.

The paper "An Effort and Time Based Measure of Usability" [13], concludes that effort and usability are related but did not address pinpointing issues. I propose to extend this work and evaluate the capability of pattern recognition techniques to pinpoint software usability issues.

In the paper "An Effort Based Model of Software Usability" [10], the authors use effort metrics to evaluate usability. Their method allows comparison of two or more implementations of the same application, but does not identify where exactly the problem lies. The approach proposed in this thesis seeks to improve on this by pinpointing usability issues.

In the paper "Classification of Usability Problems Scheme," [21] the authors describe the design and test of a defect classification scheme that extracts information from usability problems, but is limited since it does not define the causes underlying usability problems. The approach in this thesis improves on this by identifying the causes for usability problems.

In the paper “Detecting Low Usability Web Pages using Quantitative Data of Users Behavior” [22], the authors investigate the relations between quantitative data, viewing behavior of users, and web usability evaluation by subjects. They conclude that the moving speed of the gazing points is effective in detecting low usability Web pages. The research defined in this paper, however, does not point out detailed problems in a Web page. My approach tries to clearly address this issue.

In the paper “Webtracer: A New Integrated Environment for Web Usability Testing” [23], the authors used a WebTracker to evaluate usability. WebTracer is an integrated environment for web usability testing that collects the operation log of users on the Web pages. The data collected is used to determine the usability of the Web pages. However, the reasons for low usability are not identified using this approach.

CHAPTER 4

EXPERIMENT SETUP

4.1 Test Environment

Platform

The experiments in this thesis were performed on a hardware platform using an Intel® Core™ 2 Quad CPU @2.66-Ghz with 64-bit Microsoft Windows 7 Operating system.

Display Devices

A standard monitor is used for display with the subject seated in front of it.

Manual Input devices

The subject performs the tasks on the computer using the standard keyboard and mouse as input devices. An event driven logging program is used to obtain details of mouse and keystroke activities from the operating system event queue. The program saves each event along with a time stamp into a file. The logged events are: mickeys (mouse pixels), keystrokes, mouse button clicks, mouse wheel rolling, and mouse wheel clicks.

Eye Tracker

The eye tracker used for the experiments is Tobii X120 Eye Tracker, version 2.2.5. The Tobii is a standalone eye tracking unit designed for eye tracking studies. It measures unfiltered and spontaneous human reactions, responses along with gaze and other real-time data. The data collected by the eye tracker is logged to a file, which is

referred to as a *data file* in this thesis. The eye tracker also records video version of the user interaction session and is referred to as a video file which is very helpful in verifying experiment results.

4.2 Software Environment for Analysis

A software program developed in MATLAB version 7.10.0.99, 32-bit (Windows) is used to perform data analysis of the experiments performed in this thesis.

4.3 Test Procedure

Experiments conducted to evaluate the capability of pattern recognition techniques to identify software usability issues are done using the steps discussed below. Figure 4 gives the sequence of actions performed to carry out the experiments.

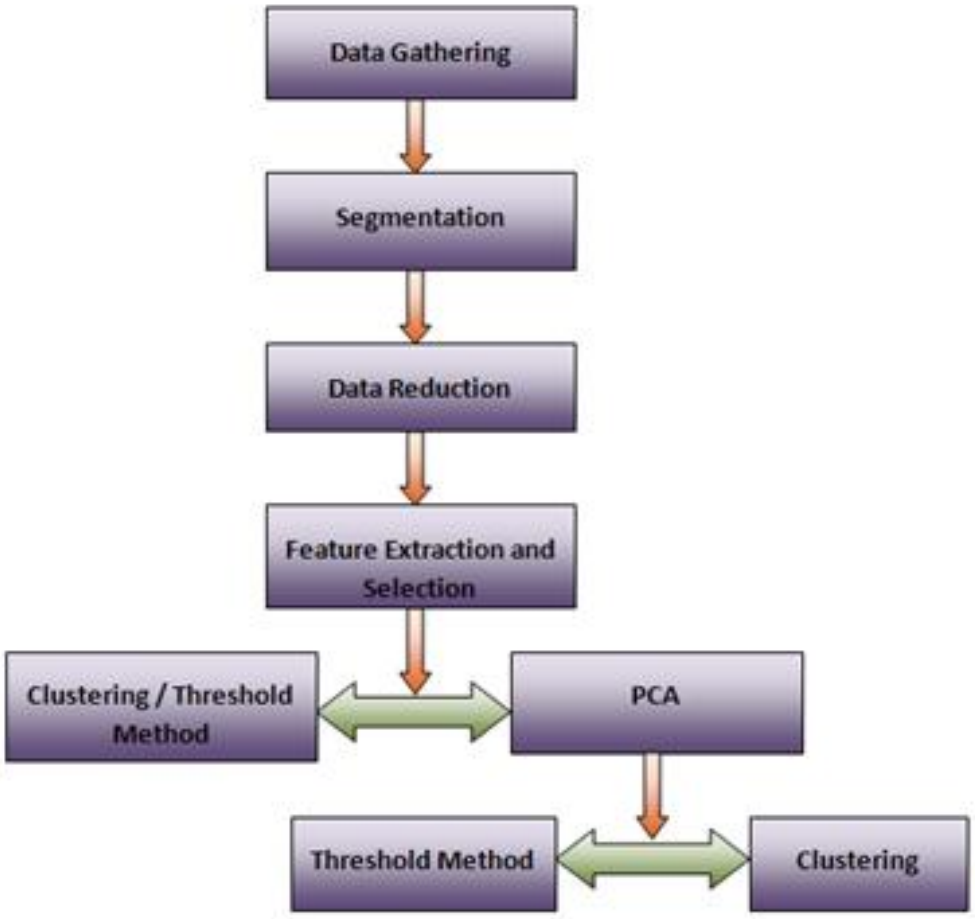

Figure 4 - Experiment Procedure

The sequence of actions in figure 4 is grouped into the following three phases:

Phase 1 - Data Gathering

The measurement process described in section 2.3.1 is employed to design the tasks and conduct the experiments. To recapitulate, a group of five users executes a set of seven identical independent tasks, which emerge from a single scenario. Throughout the

interaction process, certain user activities like eye movement, time on task, keyboard, and mouse activities are logged using an eye tracking device. According to the learnability based usability model, the point at which the user's effort reaches the acceptable level is called the learning point. Based on this model it is assumed that the user's effort reaches the acceptable level by the time they perform task 5. Hence, in this thesis, task 5 of each subject is used for conducting experiments.

Phase 2 – Data Reduction

Phase 2 includes activities such as segmentation, data reduction, and feature extraction. The data logged throughout the user interaction session, i.e., the data file is used for event based segmentation where the events are consecutive keyboard/mouse clicks. Metrics such as (a) segment duration (for event based segmentation), (b) the average fixation duration, (c) the average saccade amplitude, (d) the number of fixations, (e) the number of saccades , (f) the standard deviation of the fixation duration, (g) the standard deviation of the saccade amplitude, and (h) the eye path distance traversed is inferred for each segment. These metrics are used to generate the feature set, which is obtained by applying data reduction programs to the data file. A brief diagrammatic representation of the data reduction process is given in figure 5.

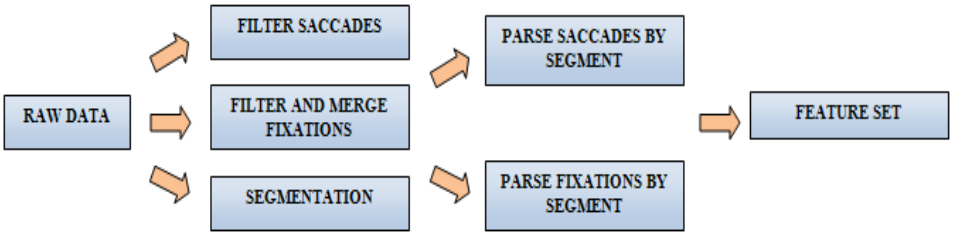

Figure 5 - Data Reduction Process

The raw data set, i.e., the data file is processed using a data reduction program developed in MATLAB. The data file is processed so that each time increment is classified as a saccade or fixation. Then saccades are processed to eliminate micro saccades and other non-interesting saccades. Fixations are processed to group and merge small fixations onto bigger fixations. Event data is analyzed and event segments are generated. Next, saccades and fixations are parsed into the events and their boundaries are adjusted. Feature data is calculated for all features within each segment and this data is useful to identify excessive effort segments.

Phase 3 – Identification of Excessive Effort Segments

Pattern recognition techniques are applied to the feature set obtained from the reduction process to identify segments that exhibit excessive effort. Identification of these excessive effort segments is automated by a program written in MATLAB. This program is referred as the *Software Program* in this thesis and the classification is referred as automatic classification. The software program takes the feature data of each segment obtained from the above data reduction process and classifies them as excessive or non-

excessive effort segments. The techniques used and applied on the feature set are briefly explained below.

In the threshold technique, a threshold value is calculated for each feature in the feature set. For a given feature, all the segments that have a feature value that is less than the threshold value are classified as non-excessive segments and vice-versa.

In the case of K-means technique, the segments are grouped into clusters. Based on the value of cluster centers, the cluster is classified as excessive or non-excessive. All segments that fall in the excessive cluster are segments exhibiting excessive effort behavior and vice versa. In the case of PCA, the first, the second, and the third principal components are obtained for the feature data. The threshold classification is applied on the first principal component and K-means clustering is applied on the first, second, and third components to classify the segments into excessive or non-excessive.

By the end of phase 3, the excessive effort segments are identified by the software program. To verify the results, the video file is carefully watched segment by segment and classified into excessive or non-excessive segments manually. The manual classification process of the video file is described in detail in the following section.

4.4 Manual Classification

The manual classification process involves event based segmentation on the entire video file. Each segment is carefully watched and classified into the following categories:

Idle behavior segments; idle behavior is due to system response. Subject waiting for a progress bar to complete or for a page to load are examples of idle behavior. Segments with such behavior are classified as idle behavior segments.

Excessive effort segments; segments without any useful user actions are classified as excessive effort segments. A subject looking at different components on an interface instead of the actual target component which help in accomplishing the task is an example for excessive effort behavior. Such behavior can be eliminated without sacrificing task completion quality.

Non-Excessive effort segments; segments with useful action that result in task completion are classified as non-excessive segments.

Off screen behavior segments: Intervals of time where the subject's view is not within the screen dimensions for more than one second, with no meaningful user action are classified as off screen behavior segments.

*Attention segments; s*egments with frequent off screen behavior, frequent mouse/keyboard clicks are classified as attention segments.

Once the video file is classified into one of the above five segment categories, the manual classification results are ready for comparison with the automatic classification results.

4.5 Verification of Results

Figure 6 is a sample result file with segment start and end times, manual classification results, and automatic classification results all consolidated into a single file. In this file, the results are obtained by applying the threshold method on one of the features, *number of fixations*, from the feature set. In figure 6, A, NE, and E denote attention, non-excessive and excessive segments respectively.

Number of Fixations			
Segment Start Time	Segment End Time	Manual Classification	Tool Classification
0	551	NE	NE
551	1451	NE	E
1451	3088	NE	E
3088	5640	NE	E
5640	5640	NE	NE
5640	10880	E	E
10880	11296	NE	NE
11296	11488	NE	NE
11488	11681	NE	NE
11681	11921	NE	NE
11921	17840	NE	NE
17840	17840	NE	NE
17840	20921	NE	E
20921	22670	A	E
22670	22670	A	NE
22670	28409	A	E
28409	30090	A	E
30090	31731	A	E
31731	33722	A	E
33722	37232	A	E
37232	37584	A	NE
37584	37728	A	NE
37728	37904	A	NE
37904	38416	A	NE
38416	40892	NE	NE

Figure 6 - Sample Result file with manual and automatic classification results

The result file in figure 6 is used to compare manual and software program results for each segment. The abbreviations used in the comparison of manual and automatic classification results are:

Excessive vs. Excessive (E vs. E); denotes that the manual classification of a particular segment is showing excessive effort and the classification of the same segment by the software program is exhibiting excessive effort.

Excessive vs. Non-Excessive (E vs. NE); denotes that the manual classification of a particular segment is showing excessive effort and the classification of the same segment by the software program is showing non-excessive effort.

Non-Excessive vs. Excessive (NE vs. E); denotes that the manual classification of a particular segment is showing non-excessive effort and the classification of the same segment by the software program is showing excessive effort.

Non-Excessive vs. Non-Excessive (NE vs. NE); denotes that the manual classification of a particular segment is showing non-excessive effort and the classification of the same segment by the software program is also showing non-excessive effort.

Attention (A); denotes segments that could not be clearly distinguished as excessive effort or non-excessive effort during manual classification.

The number of Excessive vs. Excessive, Excessive vs. Non-Excessive, Non-Excessive vs. Excessive and Non-Excessive vs. Non-Excessive segments are calculated for each result file and graphs are plotted to calculate errors and compare the results between different features. During the verification of results, the attention segments are not considered as they are not clearly distinguished as excessive effort or non-excessive effort during manual classification. Non-Excessive vs. Excessive segments are regarded as false positive or type-I error segments. It is assumed that all the segments classified as *excessive effort segments* are due for manual evaluation. Hence, in the case of type-I error, the software program is highlighting extra segments for further review, but is not missing any segments that need attention. On a similar note, segments that show excessive effort per manual classification but identified as non-excessive effort segments

by the software program are regarded as false negative or type-II error segments. These need extra attention as the software program missed identifying segments that require manual inspection. The total time of segments classified as excessive by the software program is also referred as inspection time. It is the sum of the time interval of each excessive effort segment. In this thesis, type-II errors and inspection time are considered as the most important factors for analyzing experiment results.

4.6 Experiments

In this thesis, the automatic part of the process is used to analyze five data files by applying the different pattern recognition techniques discussed. The following is a listing of the experiments performed:

1. Identifying excessive effort segments using the threshold technique
2. Identifying excessive effort segment using K-means clustering
3. Identifying excessive effort segments using the threshold technique applied to the first principal components
4. Identifying excessive effort segments using K-means clustering on first, second, and third principal components.

Each experiment procedure is discussed in detail in the following sections.

4.6.1 Experiment 1: Identifying excessive effort segments using the threshold method

In this experiment, event based segmentation is applied to the video and data file generated by the eye tracker. Next, a feature set is generated for the data file. All the

segments are classified into excessive or non-excessive effort segments by the software program which applies the threshold method on the following features: 1) number of fixations, 2) average fixation duration, 3) number of saccades, 4) average saccade amplitude, and 5) eye path traversed. Figure 7 is a diagrammatic representation of the sequence of steps followed in identifying excessive effort segments using the threshold method.

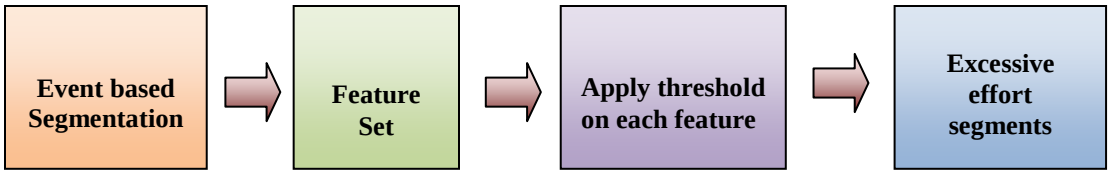

Figure 7 - Sequence of steps for identifying excessive effort segments using the threshold method

After the steps described in figure 7 are used for identifying the excessive effort segments, the video file is manually classified into excessive or non-Excessive segments based on the specifications mentioned in section 4.4. The effort segments identified through the software program and manual process are verified using five data files and their corresponding video files.

4.6.2 Experiment 2: Identifying excessive effort segments using heuristic feature selection and K-means clustering.

In this experiment, event based segmentation is applied to the video and data file generated by the eye tracker. Next, a feature set is generated for the data file. Because,

evaluating all the possible subsets of the feature set is prohibitively time consuming I have adopted the heuristic feature selection method. The following subsets are selected:

1) Number of fixations

2) Number of saccades

3) Eye path traversed

4) Number of fixations, number of saccades, eye path traversed

5) Number of fixations, number of saccades, eye path traversed, average fixation duration and average saccade amplitude. Figure 8 is a diagrammatic representation of the sequence of steps followed in identifying excessive effort segments using exhaustive feature selection and K-means clustering.

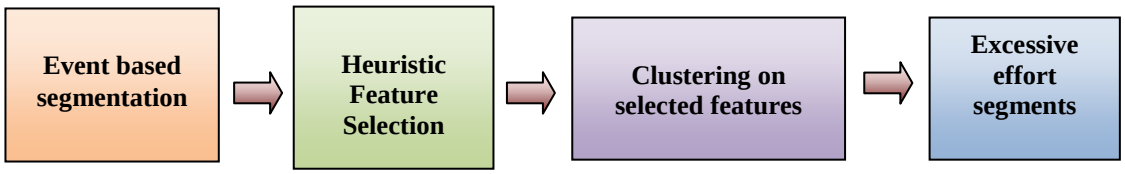

Figure 8 - Sequence of steps for identifying excessive effort segments using the K-means clustering

After the steps in figure 8 are used for identifying the excessive effort segments, the video file is manually classified into excessive or non-excessive segments based on the specifications mentioned in section 4.4. The effort segments identified through the software program and manual process are verified using five data files and their corresponding video files.

4.6.3 Experiment 3: Identifying excessive effort segments using principal component analysis

In this experiment, event based segmentation is applied to the video and data file generated by the eye tracker. Next, a feature set is generated for the data file. Following which, the feature set is transformed into principal components by a program that uses PCA function available in MATLAB. In this experiment, the first principal component is considered, as it carries the most significant information related to the feature set, and subjected to the threshold method for identifying segments exhibiting excessive effort and non-excessive effort. Figure 5.3 is a diagrammatic representation of the sequence of steps followed for identifying excessive effort segments using the threshold method on first principal component.

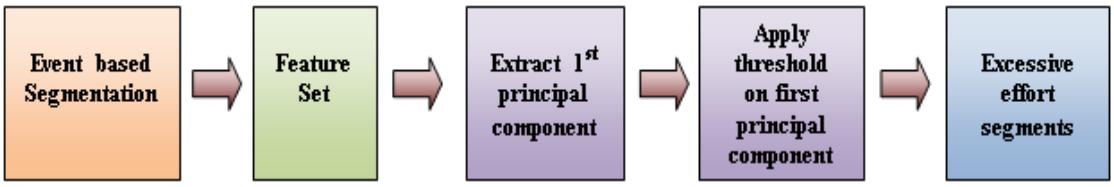

Figure 9 - Sequence of steps for identifying excessive effort segments using the threshold method on the first principal component.

After the steps discussed in figure 9 are used for identifying excessive effort segments, the video file is manually classified into excessive or non-excessive segments based on the specifications mentioned in section 4.4. The effort segments identified through the software program and manual process are verified using five data files and their corresponding video files.

4.6.4 Experiment 4: Identifying excessive effort segments using K-means clustering on principal components

In this experiment, event based segmentation is applied to the video and data file generated by the eye tracker. Next, a feature set is generated for the data file. Following, the feature set is transformed into principal components by a program that uses PCA function available in MATLAB. In this experiment, K-means clustering is applied on different combinations of principal components for identifying segments exhibiting excessive effort and non-excessive effort. The following constitute the feature set for this experiment:

1) 1st principal component
2) 1st principal component, 2nd principal component
3) 1st principal component, 2nd principal component, 3rd principal component

Figure 10 is a diagrammatic representation of the sequence of steps followed for identifying excessive effort segments using the K-means clustering on principal components.

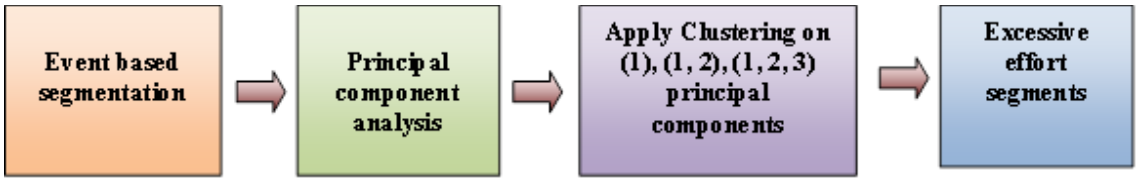

Figure 10 - Sequence of steps for identifying excessive effort segments by applying the K-means clustering on the first principal component.

Following the steps discussed in figure 10, the video file is manually classified into excessive or non-Excessive segments based on the specifications mentioned in section 4.4. The results are verified using five data files and their corresponding video files.

CHAPTER 5

EXPERIMENT RESULTS

In this chapter, the results obtained from the experiments are discussed. The results of each data file in the experiments are shown on two or three pages. All graphs are explained with a general description along with a notice regarding the local observations. The notations used for the feature values in the graphs are presented below for clarity:

- # Fix – denotes number of fixations
- Avg Fix Dur – denotes average fixation duration
- # Sacc – denotes number of saccades
- Sacc Amp – denotes average saccade amplitude
- Eye Path denotes eye path traversed

5.1 Identifying excessive effort segments using the threshold method

Data file 1 Results:

The video file corresponding to data file 1 is 6.09 minutes in length. Figure 11 shows the results of an experiment using the threshold method on data file 1.

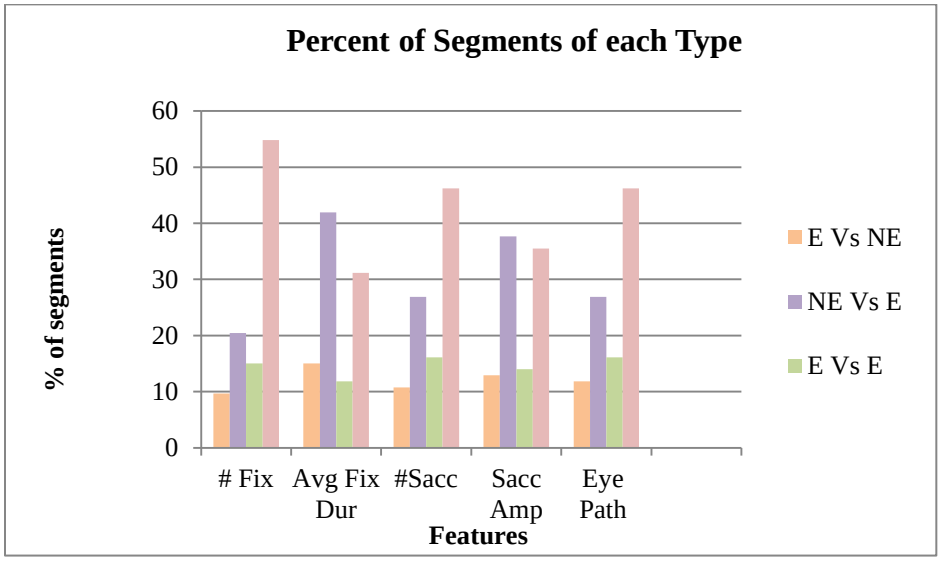

Figure 11 - Graph of percentage of segments of each type

When the graph in figure 11 is extrapolated and as seen from the orange bars, the feature value, number of fixations demonstrate a small percentage of E vs. NE segments. This shows that the number of fixations has the least number of type-II errors. Number of saccades and eye path traversed follow number of fixations in terms of type-II errors.

Figure 12 shows the total time of segments classified as excessive by the software program and the manual process after the threshold method is applied on each of the following features: 1) number of fixations, 2) average fixation duration, 3) number of saccades, 4) average saccade amplitude, and 5) eye path traversed.

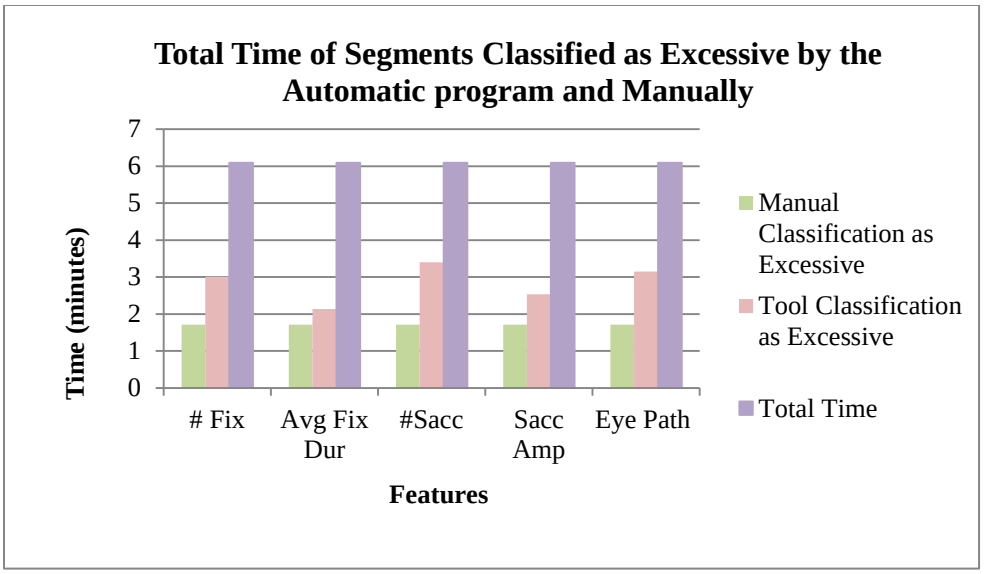

Figure 12 - Total time of segments classified as excessive by the software program and manually

The violet bars in figure 12 represent the total time of video recorded by the eye tracker. Manual classification of the video file, depicted by the green bars, shows 1.71 minutes of excessive effort. The average fixation duration and average saccade amplitude show a relatively low value for time of segments classified as excessive by the software program when compared with the total video time. This is depicted by the pink bars in the above graph. From figure 11 it is observed that the percentage of type-II errors is 15.05% for average fixation duration and 12.9% for average saccade amplitude. However, the feature value with a reasonable type-II errors and lower percentage of time of segments classified as excessive is average saccade amplitude.

Data file 2 Results:

The video file corresponding to the data file 2 is 3.27 minutes in length. Figure 13 shows the results of an experiment using the threshold method on the data file 2.

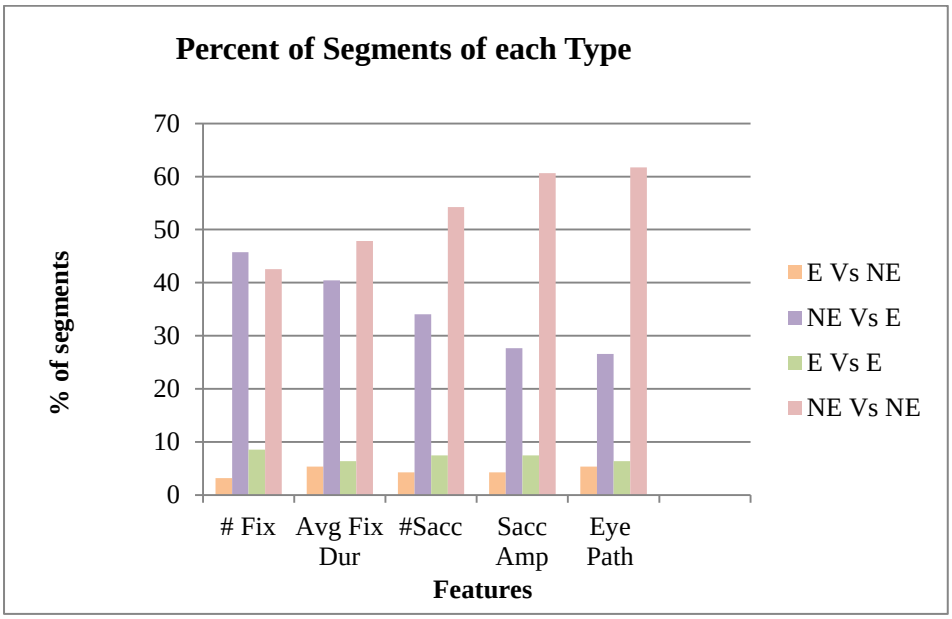

Figure 13 - Graph of percentage of segments of each type

When the graph in figure 13 is extrapolated and as seen from the orange bars, the feature value, number of fixations demonstrates a small percentage of E vs. NE segments. This shows that the number of fixations has the least number of type-II errors. Number of saccades and average saccade amplitude follow number of fixations in terms of type-II errors.

Figure 14 shows the total time of segments classified as excessive by the software program and the manual process for the features: 1) number of fixations, 2) average fixation duration, 3) number of saccades, 4) average saccade amplitude, and 5) eye path traversed.

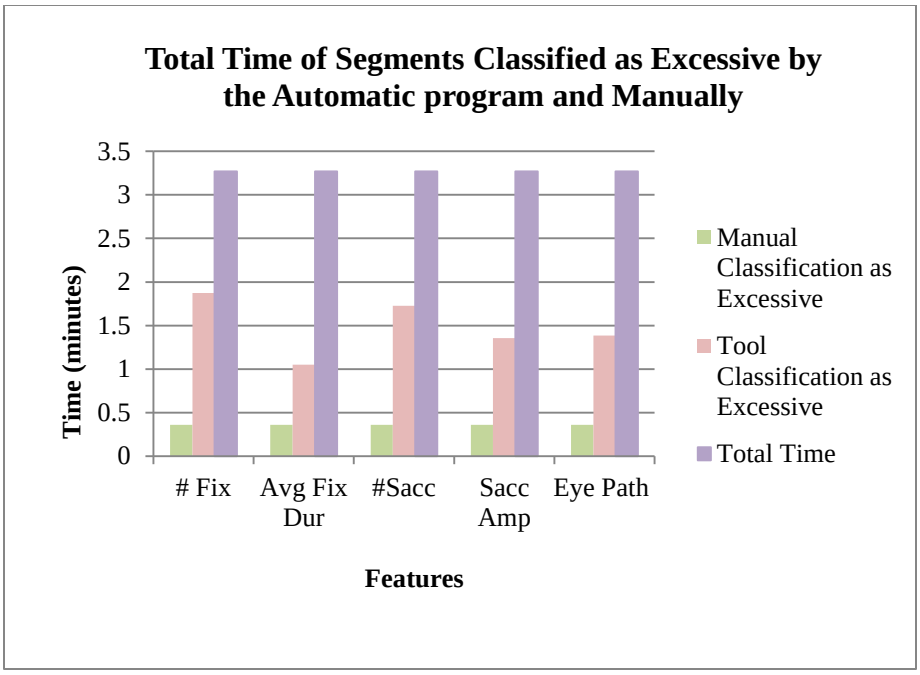

Figure 14 - Total time of segments classified as excessive by the software program and manually

The violet bars in figure 14 represent the total time of video recorded by the eye tracker. Manual classification of the video file, depicted by the green bars, shows 0.36 minutes of excessive effort. The average fixation duration and eye path traversed show a relatively low value for time of segments classified as excessive by the software program when compared with the total video time. This is depicted by the pink bars in the above graph. From figure 13 it is observed that the percentage of type-II errors is 5.319 % for both average fixation duration and eye path traversed. The feature value with an acceptable type-II errors and minimal percentage of time of segments classified as excessive is eye path traversed.

Data file 3 Results:

The video file corresponding to data file 3 is 3.8 minutes in length. Figure 15 shows the results of an experiment using the threshold method on data file 3.

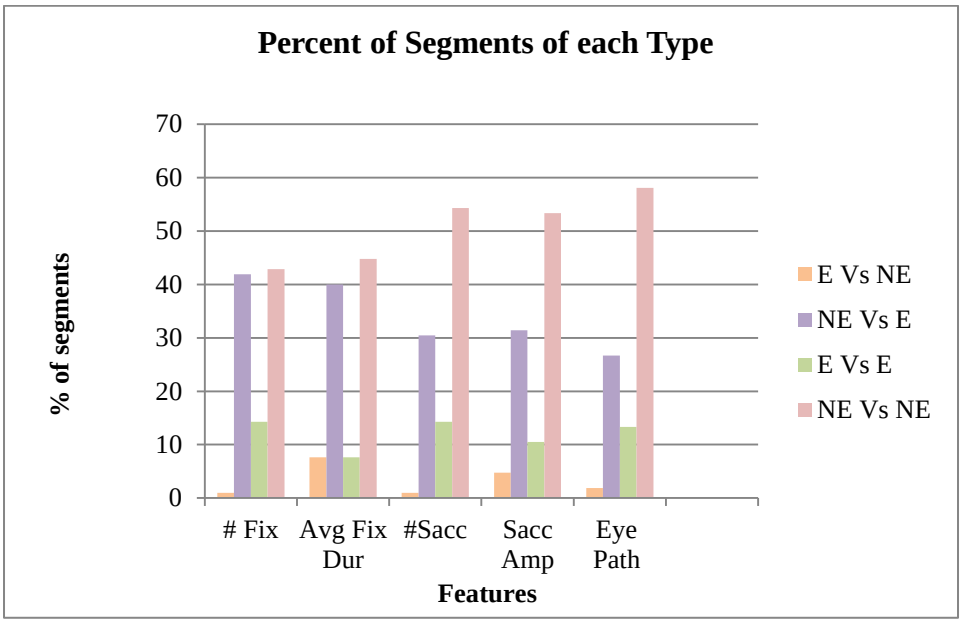

Figure 15 - Graph of percentage of segments of each type

When the graph in figure 15 is extrapolated and as seen from the orange bars, the feature value, number of saccades demonstrates a small percentage of E vs. NE segments as seen from the orange bars. This shows that the number of saccades has the least number of type-II errors. Eye path traversed and number of fixations follow number of saccades in terms of type-II errors.

Figure 16 shows the total time of segments classified as excessive by the software program and the manual process for the features: 1) number of fixations, 2) average fixation duration, 3) number of saccades, 4) average saccade amplitude, and 5) eye path travelled.

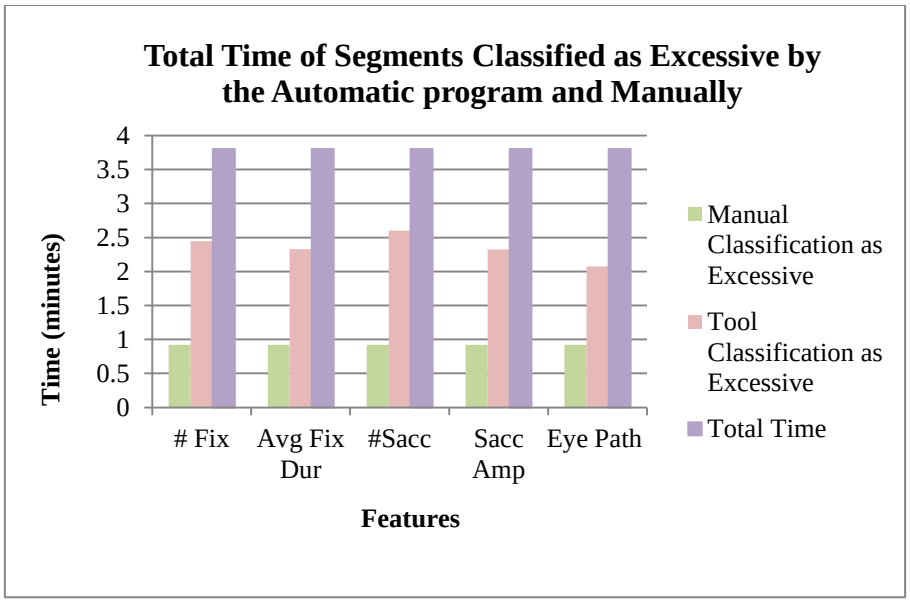

Figure 16 - Total time of segments classified as excessive by the software program and manually

The violet bars in figure 16 represent the total time of video recorded by the eye tracker. Manual classification of the video file, depicted by the green bars, shows 0.92 minutes of excessive effort. All the features considered in this experiment have a relatively close value for time of segments classified as excessive by the software program when compared with the total video time. This is depicted by the pink bars in the above graph. From figure 15 it is observed that the eye path traversed and average saccade amplitude have the least percentage of type-II errors. However, the feature value with lower type-II errors and lower percentage of time of segments classified as excessive is eye path traversed.

Data file 4 Results:

The video file corresponding to data file 4 is 3.5 minutes in length. Figure 17 shows the results of an experiment using the threshold method on data file 4.

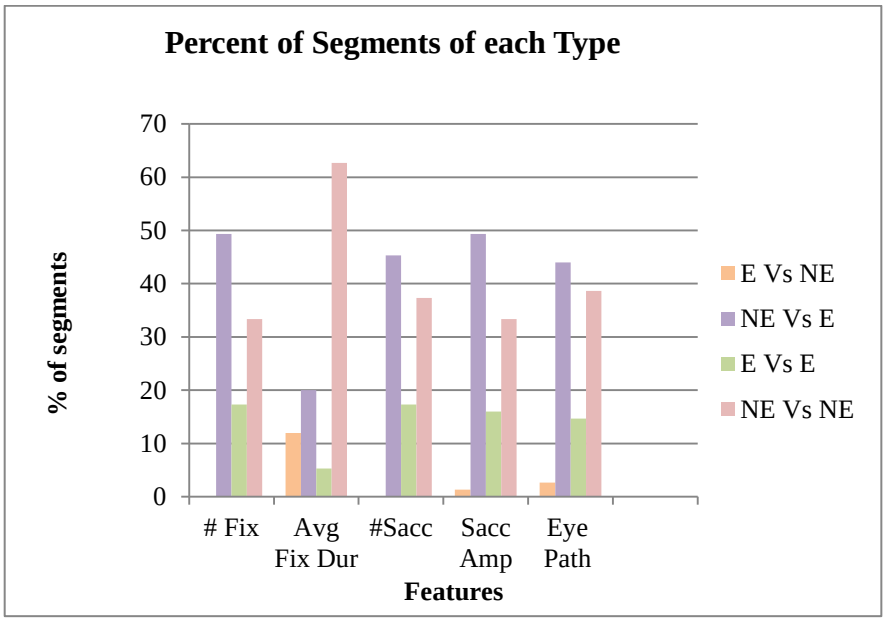

Figure 17 - Graph of percentage of segments of each type

When the graph in figure 17 is reviewed and as seen from the missing orange bars, the two features, number of fixations and number of saccades, have no E vs. NE segments. This shows that the two features have no type-II errors.

Figure 18 shows the total time of segments classified as excessive by the software program and the manual process for the features: 1) number of fixations, 2) average fixation duration, 3) number of saccades, 4) average saccade amplitude, and 5) eye path traversed.

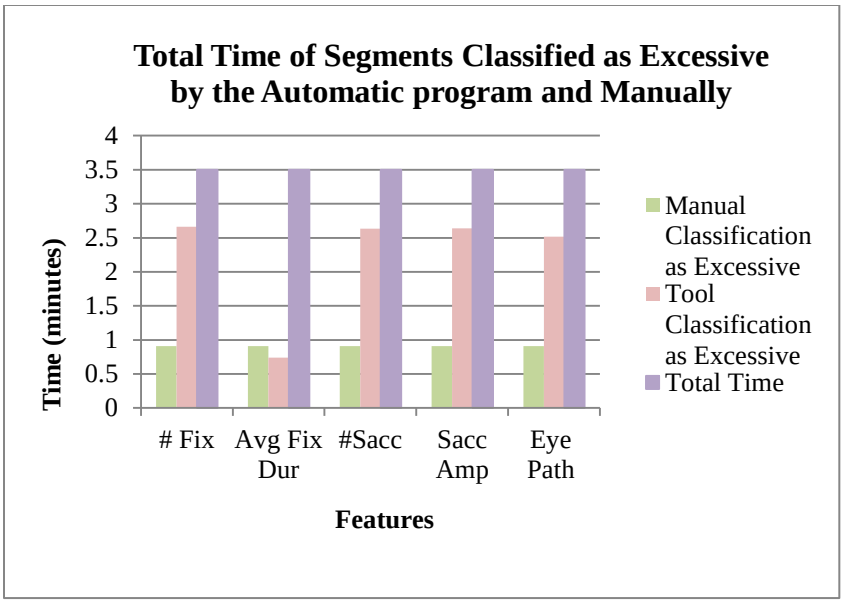

Figure 18 - Total time of segments classified as excessive by the software program and manually

The violet bars in figure 18 represent the total time of video recorded by the eye tracker which is 3.5 minutes. Manual classification of the video file, depicted by the green bars, shows 0.91 minutes of excessive effort. The average fixation duration shows a relatively low value for time of segments classified as excessive by the software program when compared with the total video time. This is depicted by the pink bars in the above graph. From figure 17 it is observed that the percentage of type-II errors for average fixation duration is 12 %. Therefore, the feature value with an acceptable error of type-II and lower percentage of time of segments classified as excessive is average fixation duration.

Data file 5 Results:

The video file corresponding to data file 5 is 5.65 minutes in length. Figure 19 shows results of an experiment using the threshold method on data file 5.

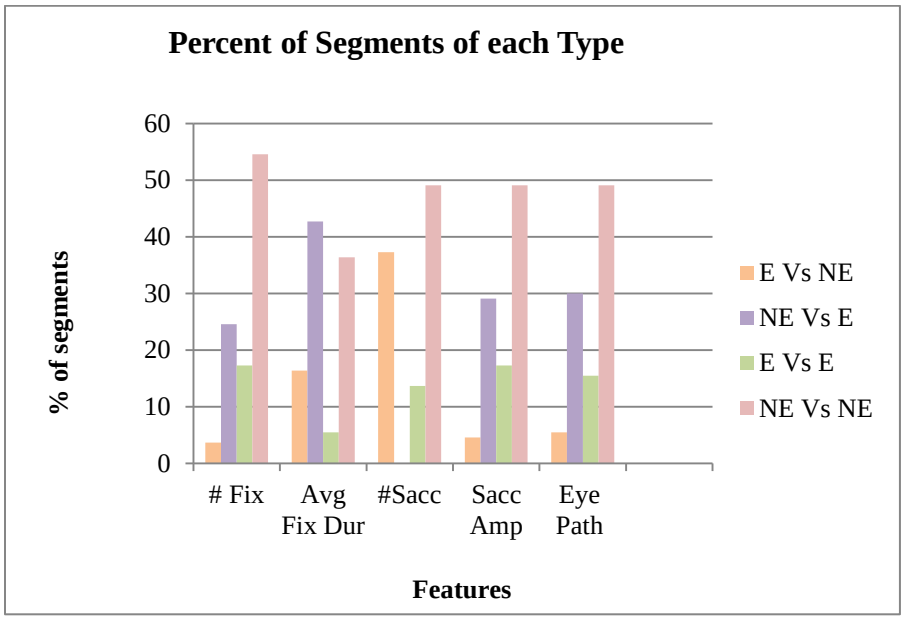

Figure 19 - Graph of percentage of segments of each type

When the graph in figure 19 is extrapolated and as seen from the orange bars, the feature value, number of fixations shows a small percentage of E vs. NE segments. This shows that the number of fixations has fewer type-II errors. Average saccade amplitude and eye path traversed follow number of fixations in terms of type-II errors.

Figure 20 shows the total time of segments classified as excessive by the software program and the manual process for the features: 1) number of fixations, 2) average fixation duration, 3) number of saccades, 4) average saccade amplitude, and 5) eye path traversed.

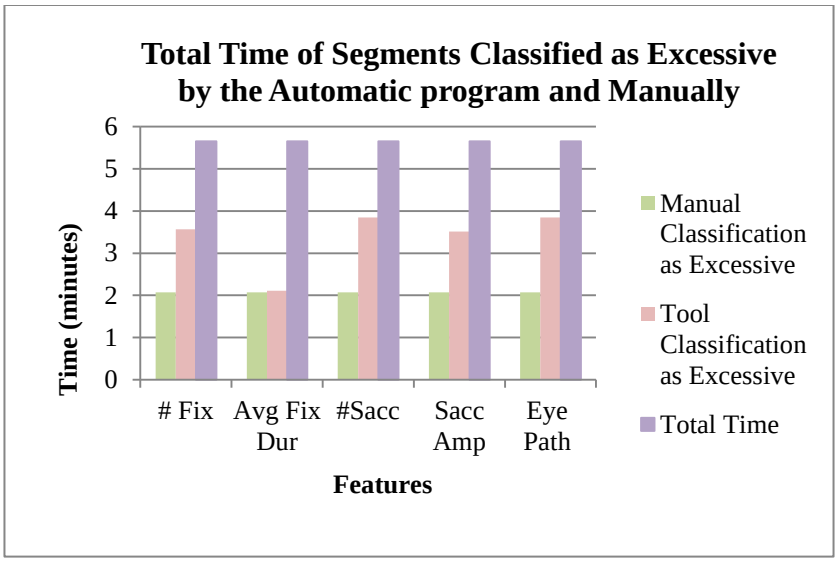

Figure 20 - Total time of segments classified as excessive by the software program and manually

The violets bars in figure 20 represent the total time of video recorded by the eye tracker and are 5.65 minutes. Manual classification of the video file, depicted by the green bars, shows 2.07 minutes of excessive effort. The average fixation duration shows a relatively low value for time of segments classified as excessive by the software program when compared with the total video time. From figure 19 it is observed that the percentage of type-II errors for average fixation duration is 16.36 %, which is not within an acceptable limit. All features other than average fixation duration have relatively close values for time of segments classified as excessive by the software program. Therefore, the feature value with an acceptable error of type-II and lower percentage of time of segments classified as excessive is number of fixations.

5.2 Identifying excessive effort segments using heuristic feature selection and K-means clustering.

Data file 1 Results:

The video file corresponding to data file 1 is 6.09 minutes in length. Figure 21 shows the results of an experiment using the K-means clustering on data file 1.

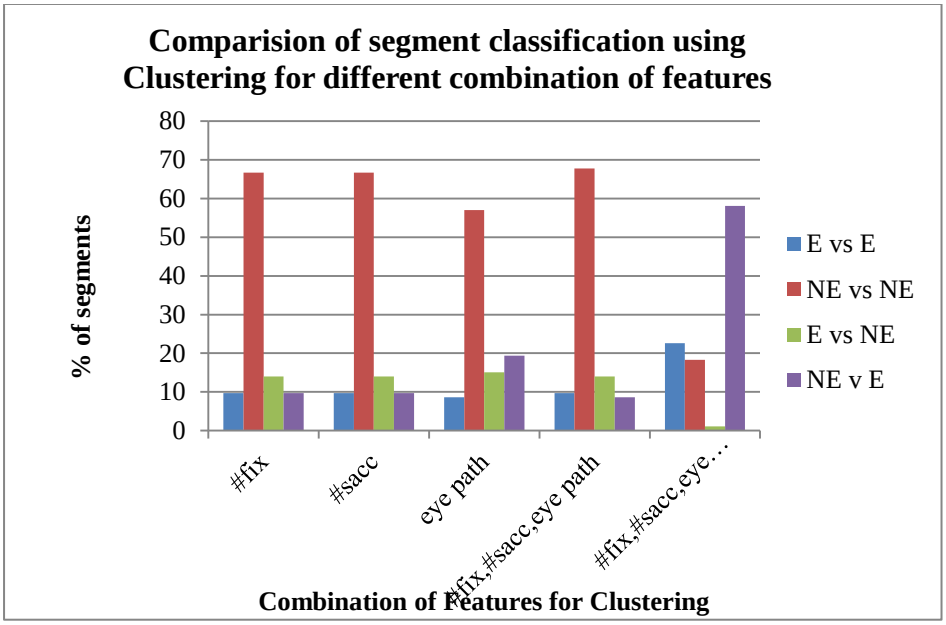

Figure 21 - Graph of percentage of segments of each type

When the graph in figure 21 is extrapolated and as seen from the green bars, the feature subset 5 demonstrates a small percentage of E vs. NE segments. This shows that the feature subset 5 has the least number of type-II errors. All the other features in this experiement have very high percentage of E vs. NE segments and are not suitable for consideration.

Figure 22 shows the total time of segments classified as excessive by the software program and the manual process after the K-means clustering is applied on the above defined feature subsets.

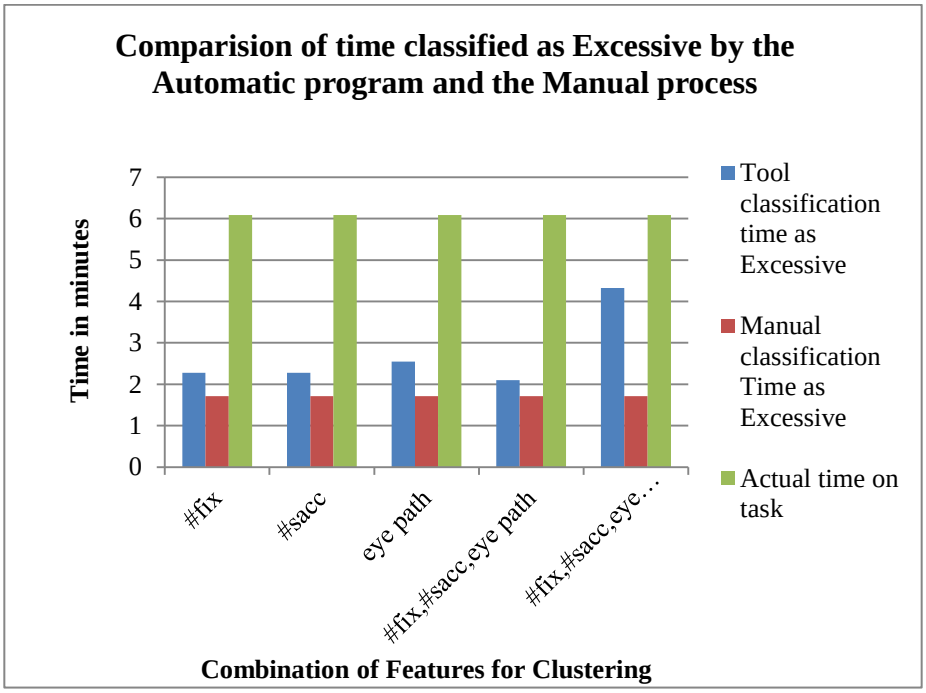

Figure 22 - Total time of segments classified as excessive by the software program and manually

The green bars in figure 22 represent the total time of video recorded by the eye tracker. Manual classification of the video file, depicted by the maroon bars, shows 1.71 minutes of excessive effort. The feature subsets 1, 2, and 4 show a relatively low value for time of segments classified as excessive by the software program when compared with the total video time. This is depicted by the pink bars in the above graph. From figure 21 it is observed that the percentage of type-II errors is 13.98% for subset 1, 13.98% for subset 2, and 13.98 % for subset4. All the feature subsets except for subset 5 have very high values for type-II errors making it hard to select a feature subset with

lower percentage of time classified as excessive by the software program, while also ensuring that the type-II errors are within an acceptable limit.

Data file 2 Results:

The video file corresponding to data file 2 is 3.27 minutes in length. Figure 23 shows the results of an experiment using the K-means clustering on data file 2.

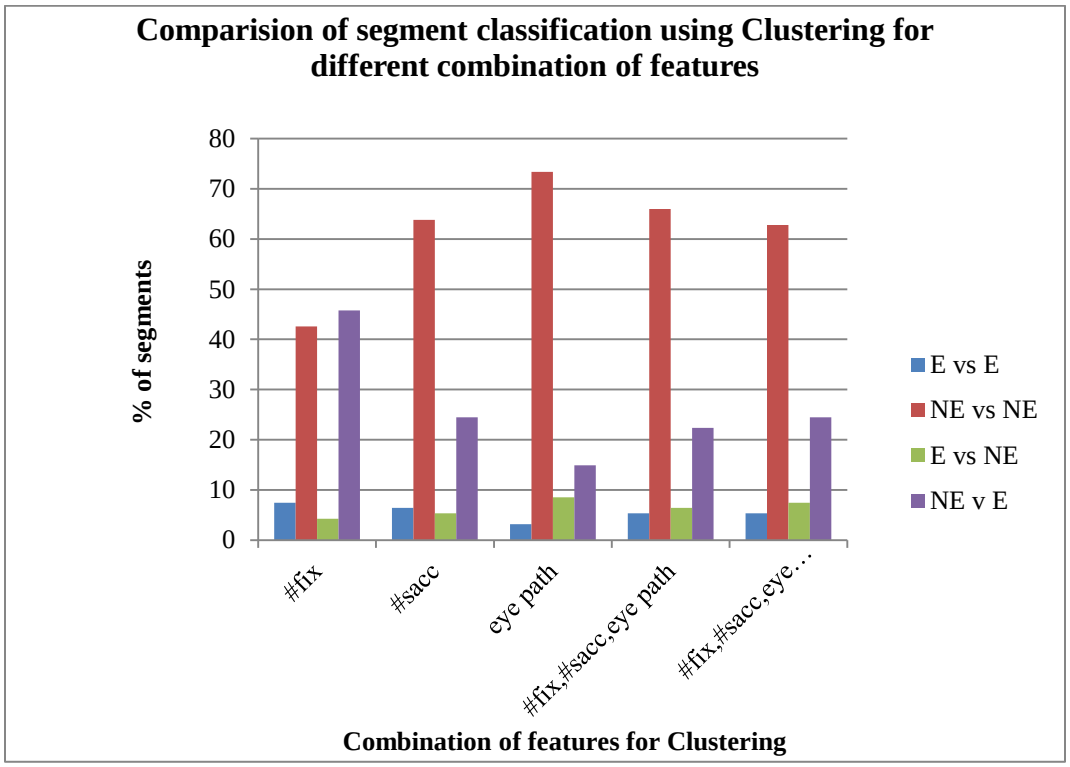

Figure 23 - Graph of percentage of segments of each type

When the graph in figure 23 is extrapolated and as seen from the green bars, the feature subset 1 demonstrates a small percentage of E vs. NE segments. This shows that the subset1 has the least number of type-II errors. Subset 2 follows subset 1 in terms of type-II errors.

Figure 24 shows the total time of segments classified as excessive by the software program and the manual process for the above defined five feature subsets.

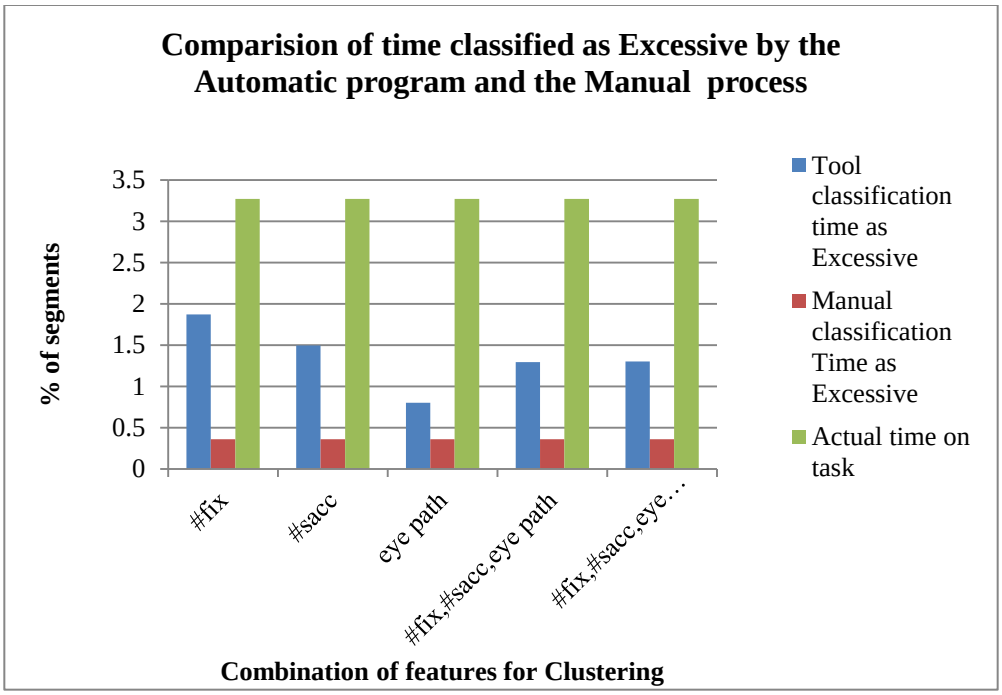

Figure 24 - Total time of segments classified as excessive by the software program and manually

The violet bars in figure 24 represent the total time of video recorded by the eye tracker. Manual classification of the video file, depicted by the maroon bars, shows 0.36 minutes of excessive effort. The subset 3 shows a relatively low value for time of segments classified as excessive by the software program when compared with the total video time. This is depicted by the pink bars in the above graph. From figure 23 it is observed that the percentage of type-II errors is 8.5% for subset 3. Therefore, the feature value with an acceptable error of type-II and lower percentage of time of segments classified as excessive is subset 3.

Data file 3 Results:

The video file corresponding to data file 1 is 3.8 minutes in length. Figure 25 shows results of an experiment using the K-means clustering on data file 3.

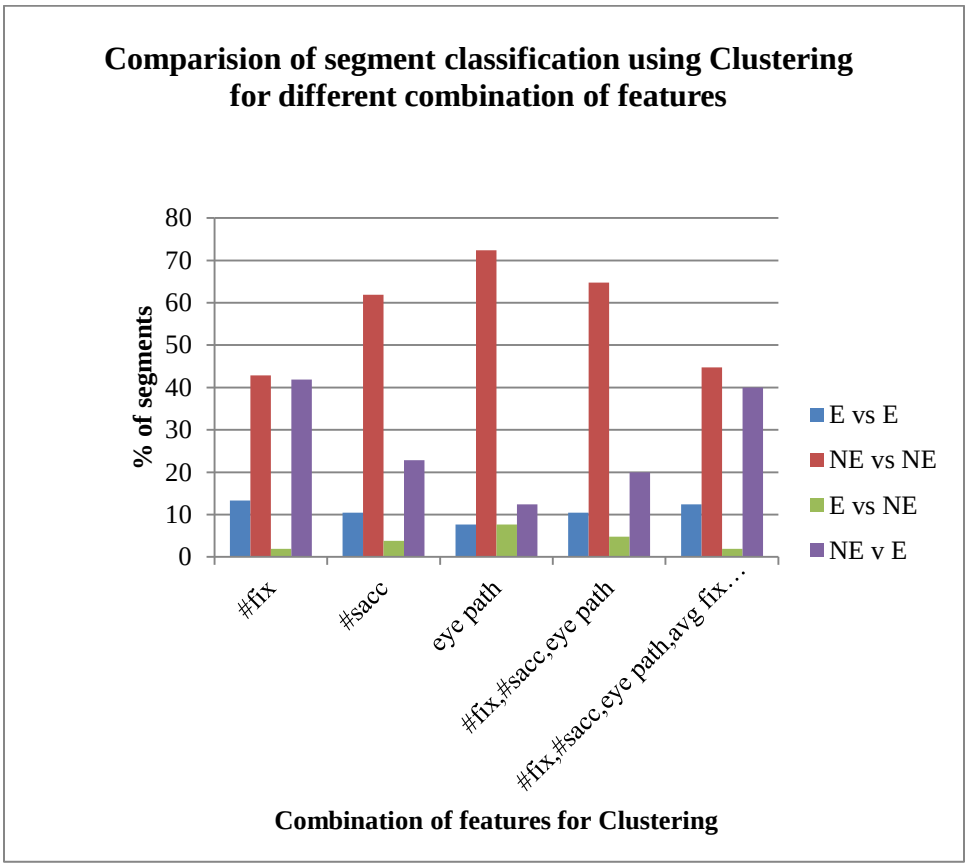

Figure 25 - Graph of percentage of segments of each type

When the graph in figure 25 is extrapolated as seen from the green bars, the two feature subsets 1 and 5 demonstrate a small percentage of E vs. NE segments. This shows that the above two feature subsets have the least number of type-II errors.

Figure 26 shows the total time of segments classified as excessive by the software program and the manual process for the above defined five feature subsets.

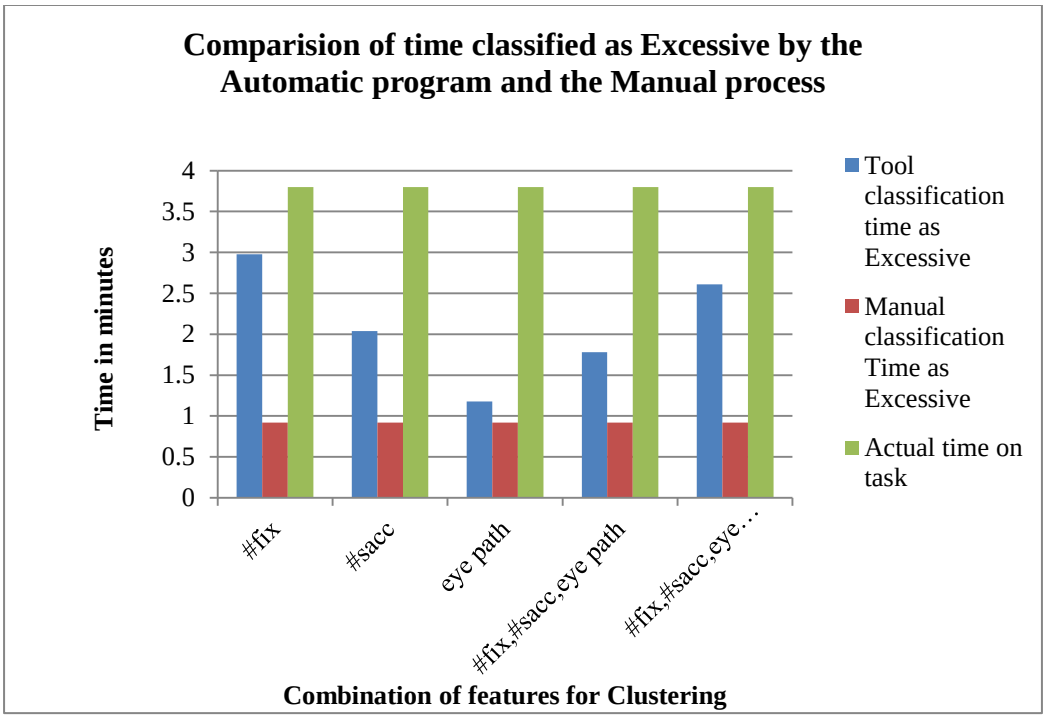

Figure 26 - Total time of segments classified as excessive by the software program and manually

The green bars in figure 26 represent the total time of video recorded by the eye tracker. Manual classification of the video file, depicted by the maroon bars, shows 0.92 minutes of excessive effort. The feature subsets 3 and 4 show a relatively low value for time of segments classified as excessive by the software program when compared with the total video time. This is depicted by the pink bars in the above graph. From figure 25 it is observed that the percentage of type-II errors is 7.69% for subset 3 and 4.76% for feature subset 4. However, the feature value with lower type-II errors and lower percentage of time of segments classified as excessive is feature subset 3.

Data file 4 Results:

The video file corresponding to data file 4 is 3.5 minutes in length. Figure 27 shows results of an experiment using the K-means clustering on the data file.

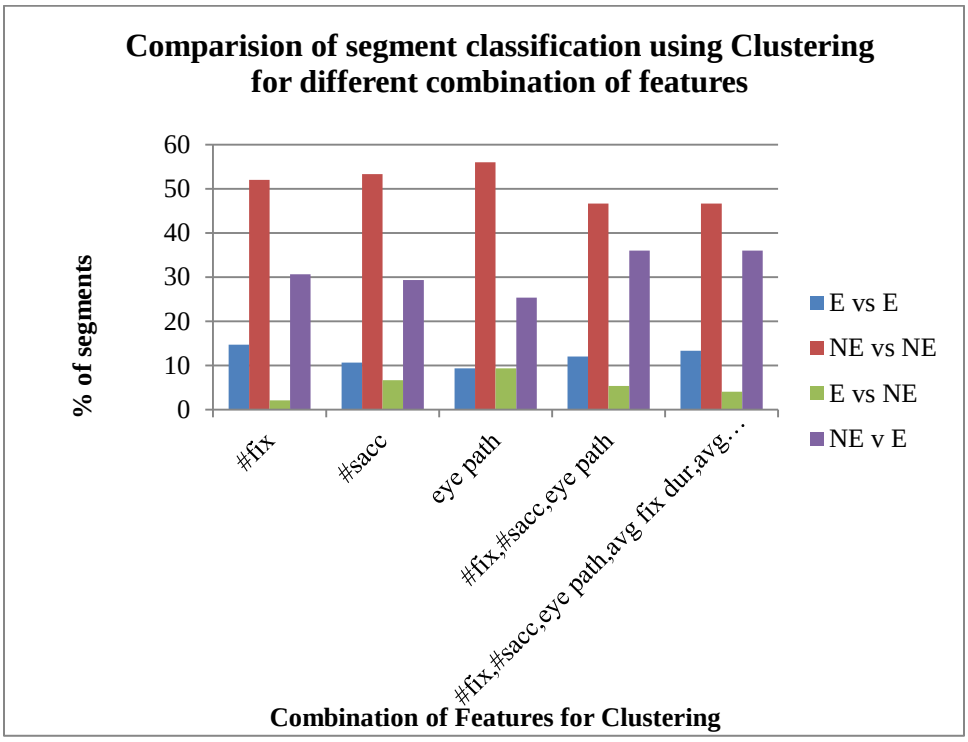

Figure 27 - Graph of percentage of segments of each type

When the graph in figure 27 is extrapolated as seen from the green bars, the two feature subset 1 has the least percentage of E vs. NE segments. This shows that the feature subset 1 has fewer type-II errors. Feature subset 5 follows subset 1 in terms of type-II errors.

Figure 28 shows the total time of segments classified as excessive by the software program and the manual process for the above defined five feature subsets.

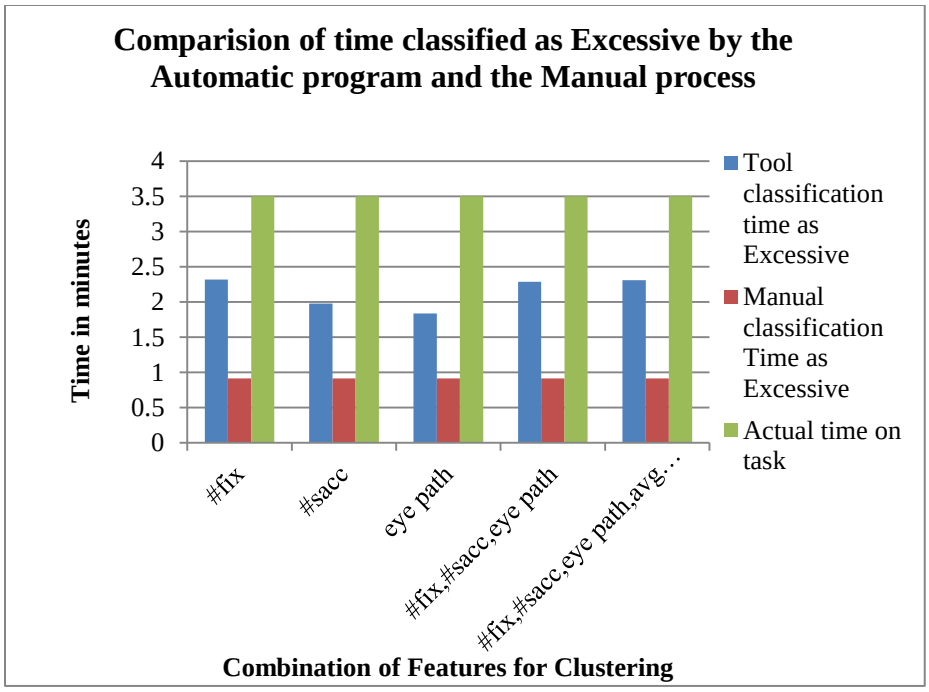

Figure 28 - Total time of segments classified as excessive by the software program and manually

The green bars in figure 28 represent the total time of video recorded by the eye tracker and are 3.5 minutes. Manual classification of the video file, depicted by the maroon bars, shows 0.91 minutes of excessive effort. The feature subsets 2 and 3 show a relatively low value for time of segments classified as excessive by the software program when compared with the total video time. This is depicted by the pink bars in the above graph. From figure 27 it is observed that the percentage of type-II errors is 6.66% for subset 2 and 9.33% for feature subset 3. However, the feature value with an acceptable error of type-II and lower percentage of time of segments classified as excessive is feature subset 2.

Data file 5 Results:

The video file corresponding to data file 5 is 5.65 minutes in length. Figure 29 shows results of an experiment using the K-means clustering on data file 5.

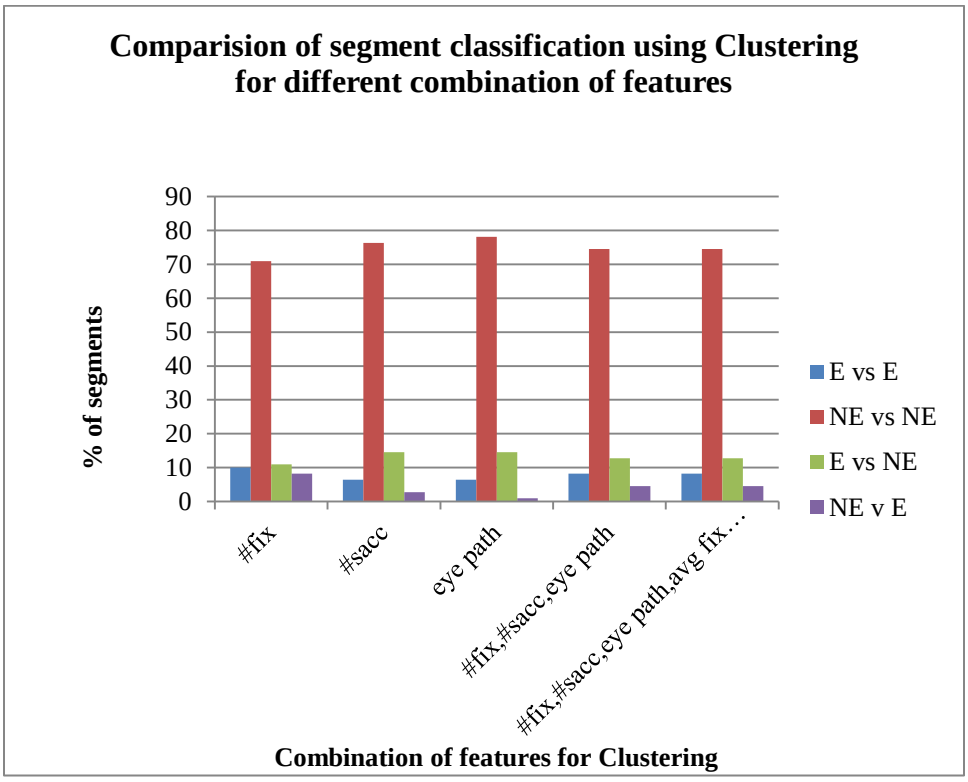

Figure 29 - Graph of percentage of segments of each type

When the graph in figure 29 is extrapolated as seen from the green bars, the feature subset 1 shows a small percentage of E vs. NE segments. This shows that the above feature set has fewer type-II errors. Feature subset 4 and 5 follow feature subset 1 in terms of type-II errors.

Figure 30 shows the total time of segments classified as excessive by the software program and the manual process for the above defined feature subsets.

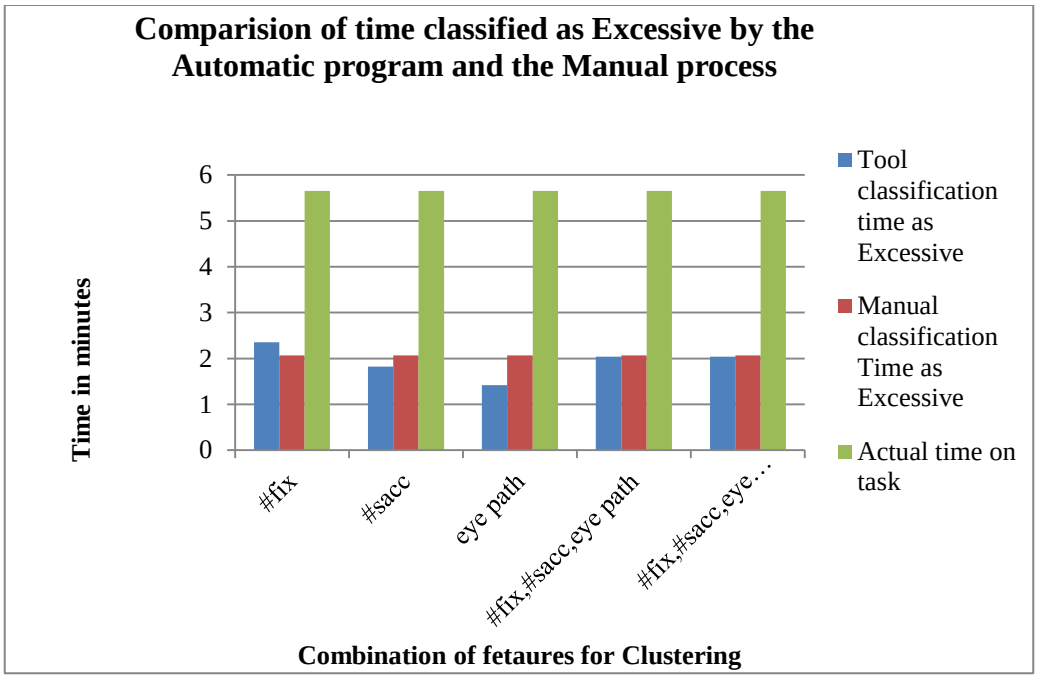

Figure 30 - Total time of segments classified as excessive by the software program and manually

The violets bars in figure 30 represent the total time of video recorded by the eye tracker and are 5.65 minutes. Manual classification of the video file, depicted by the maroon bars, shows 2.07 minutes of excessive effort. The feature subsets 2 and 3 show a relatively low value for time of segments classified as excessive by the software program when compared with the total video time. This is depicted by the pink bars in the above graph. From figure 29 it is observed that the percentage of type-II errors for subset2 and subset 3 is not within an acceptable limit. So, the feature value with an acceptable value of type-II errors and relatively low value for percentage of time of segments classified as excessive is feature subset 4 or subset 5.

5.3 Identifying excessive effort segments using principal component analysis

The results of all the data files are consolidated into a single graph. Figure 31 shows the percentage of segments of each type when applying the threshold method on the first principal component for all five data files.

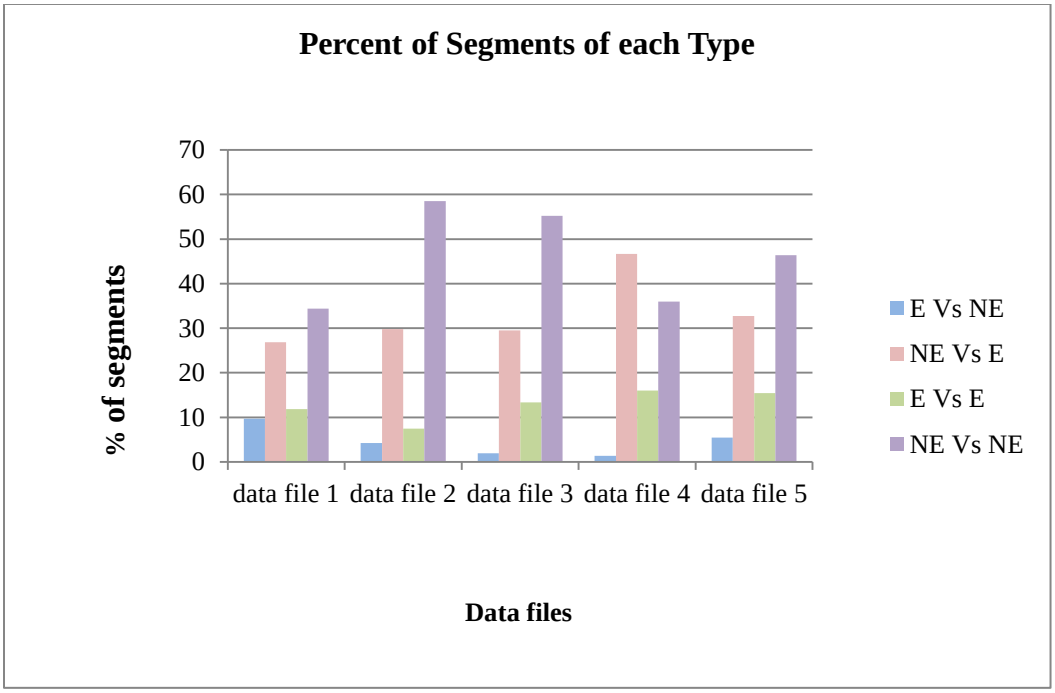

Figure 31 - Percentage of segments of each type when applying the threshold method on first principal component

From the graph in figure 31 it is clear that using the threshold method on the first principal components produces a small percentage of E vs. NE segments. This means lower type-II errors as seen from the blue bars in the above graph. Figure 32 shows the total time of segments classified as excessive by the software program and the manual

process for the feature value, first principal component. The results of all five data files are plotted in a single graph.

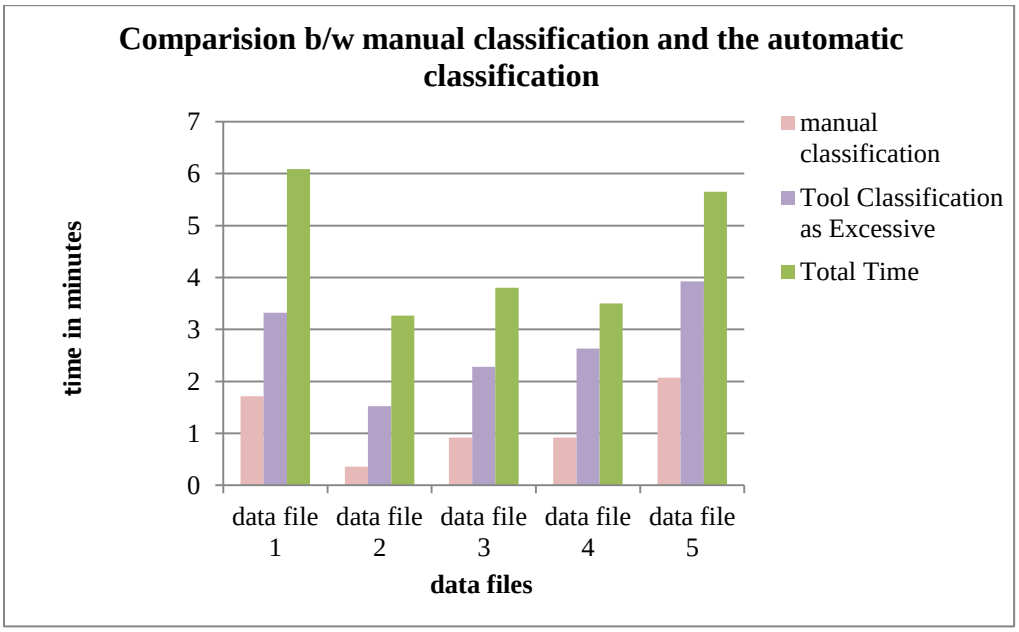

Figure 32 - Total time of segments classified as excessive by the software program and manually

The green bars in Figure 32 represent the total time of video recorded by the eye tracker for all five data files. Manual classification of the video files is depicted by the pink bar. The violet bars represent the total time of video classified as excessive by the software program. The percentage of time of segments classified as excessive is relatively high when applying thresholding on first principal component.

5.4 Identifying excessive effort segments using K-means clustering on principal components.

Data file 1 Analysis:

The video file corresponding to data file 1 is 6.09 minutes in length. Figure 33 shows the percentage of E vs. E, E vs. NE, NE vs. NE and NE vs. E segments for the features mentioned in the experiment description.

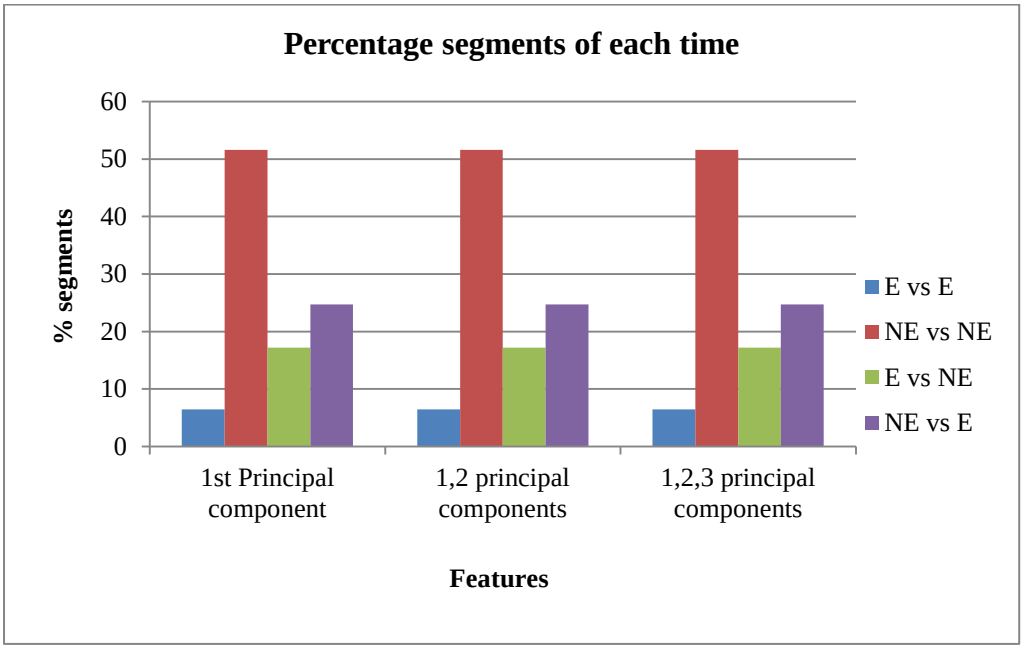

Figure 33 - Graph of percentage of segments of each type

From the graph in figure 33 it is clear that all the three features have the same percentage of E vs. E, E vs. NE, NE vs. NE and NE vs. E segments. This means that all the feature values have the same percentage of type-I and type-II errors.

Figure 34 shows the total time of segments classified as excessive by the software program and the manual process for the three features.

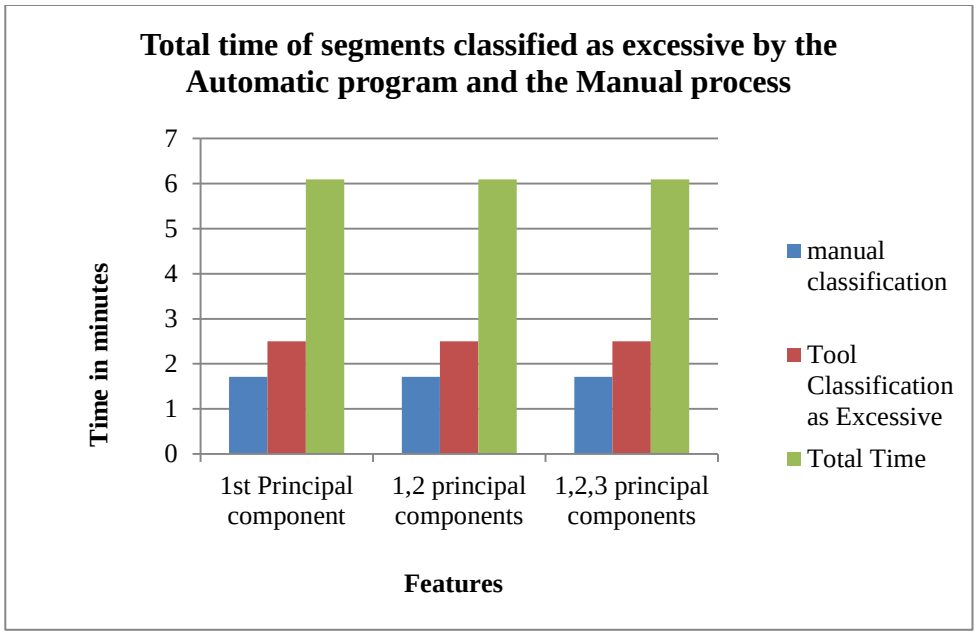

Figure 34 - Total time of segments classified as excessive by the software program and manually

The green bars in figure 34 represent the total time of video recorded by the eye tracker. Manual classification of the video file, depicted by the blue bars, shows 1.71 minutes of excessive effort. The automatic classification of the video file for the three features shows 2.5 minutes of excessive effort time which is depicted by the maroon bars in figure 34. However, the type-II errors are very high and not within an acceptable limit.

Data file 2 Analysis:

The video file corresponding to data file 2 is 3.27 minutes in length. Figure 35 shows the percentage of E vs. E, E vs. NE, NE vs. NE and NE vs. E segments for the features mentioned in the experiment description.

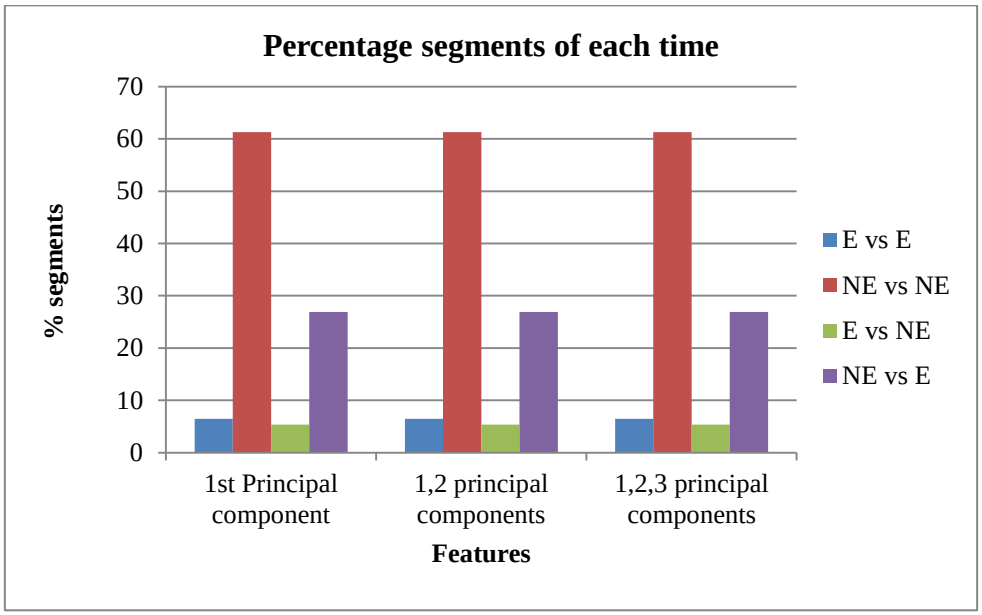

Figure 35 - Graph of percentage of segments of each type

From the graph in figure 35 it is clear that all the three features have the same percentage of E vs. E, E vs. NE, NE vs. NE and NE vs. E segments. This means that all the feature values have the same percentage of type-I and type-II errors.

Figure 36 shows the total time of segments classified as excessive by the software program and the manual process for the three features.

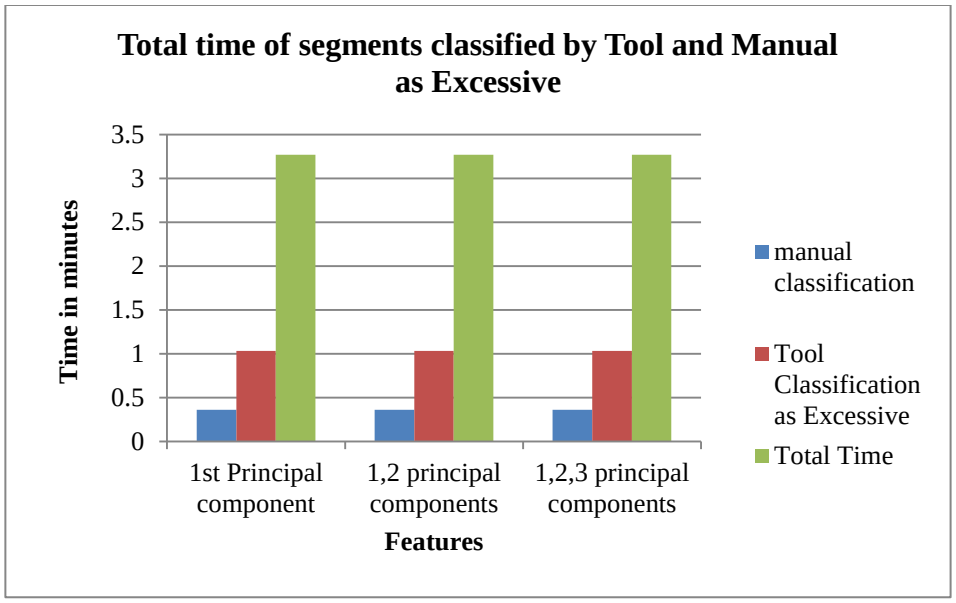

Figure 36 - Total time of segments classified as excessive by the software program and manually

The green bars in figure 36, represent the total time of video recorded by the eye tracker. Manual classification of the video file, depicted by the blue bars, shows 0.36 minutes of excessive effort. The automatic classification of the video file for all the three features shows 1.03 minutes of excessive effort time which is depicted by the maroon bars in figure 36. All the features have an acceptable value of type-II errors at 5.37%.

Data file 3 Results:

The video file corresponding to data file 3 is 3.8 minutes in length. Figure 37 shows the percentage of E vs. E, E vs. NE, NE vs. NE and NE vs. E segments for the features mentioned in the experiment description.

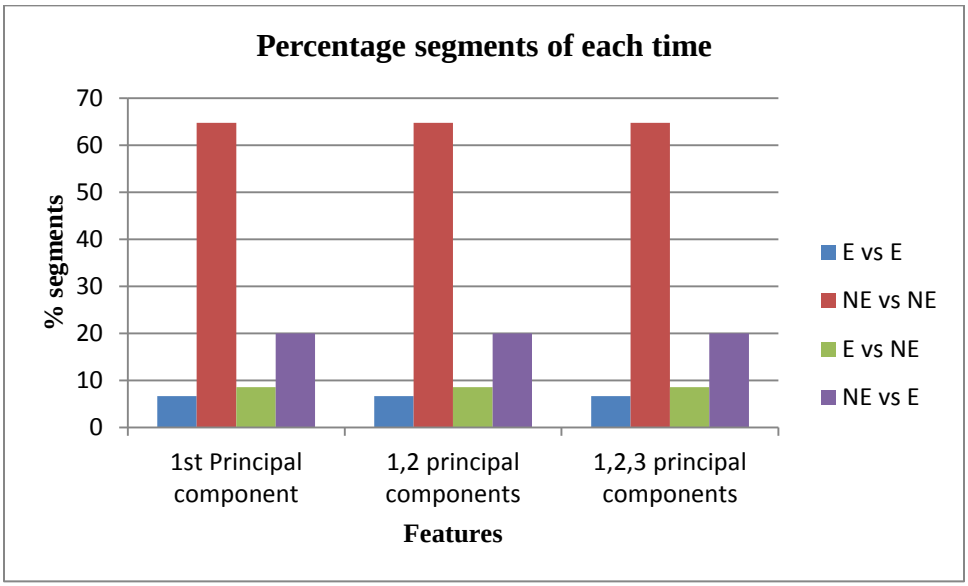

Figure 37 - Graph of percentage of segments of each type

From the graph in figure 37 it is clear that all the three features have same percentage of E vs. E, E vs. NE, NE vs. NE and NE vs. E segments . This means that all the feature values have the same percentage of type-I and type-II errors.

Figure 38 shows the total time of segments classified as excessive by the software program and the manual process for the three features.

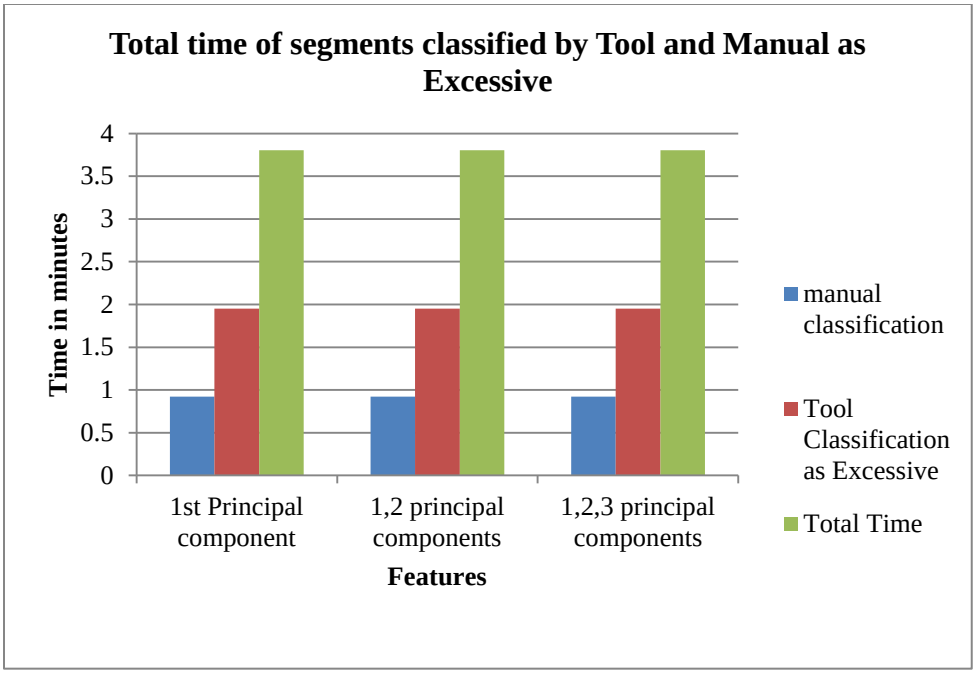

Figure 38 - Total time of segments classified as excessive by the software program and manually

The green bars in figure 38 represent the total time of video recorded by the eye tracker. Manual classification of the video file, depicted by the blue bars, shows 0.92 minutes of excessive effort. The automatic classification of the video file for all the three features shows 1.95 minutes of excessive effort time which is depicted by the maroon bars in figure 38. All the features have an acceptable value of type-II errors at 8.57%.

Data file 4 Results:

The video file corresponding to data file 4 is 3.5 minutes in length. Figure 39 shows the percentage of E vs. E, E vs. NE, NE vs. NE and NE vs. E segments for the features mentioned in the experiment description.

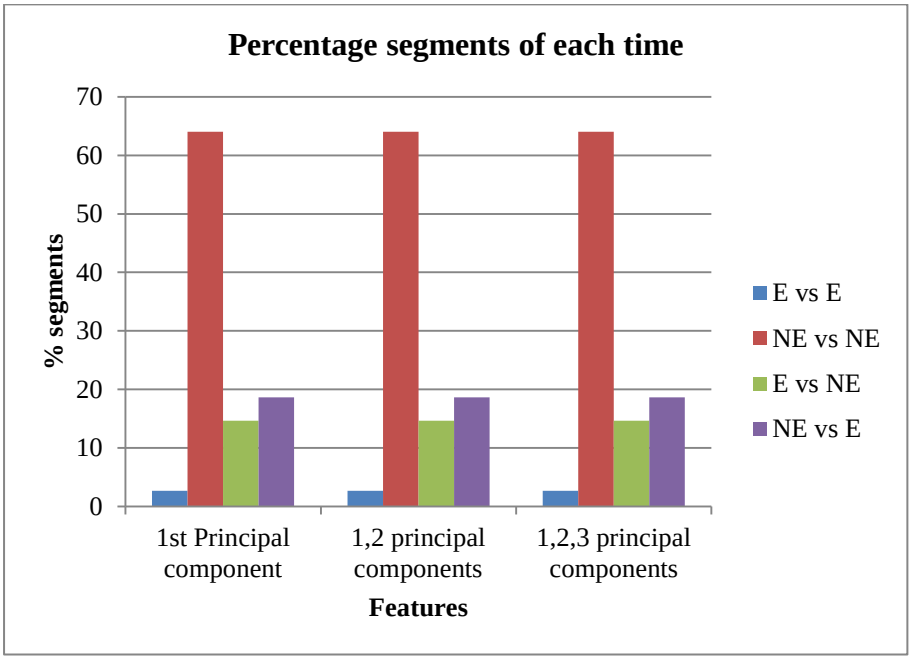

Figure 39 - Graph of percentage of segments of each type

From the graph in figure 39 it is clear that all the three features have the same percentage of E vs. E, E vs. NE, NE vs. NE and NE vs. E segments. This means that all the feature values have the same percentage of type-I and type-II errors.

Figure 40 shows the total time of segments classified as excessive by the software program and the manual process for the three features.

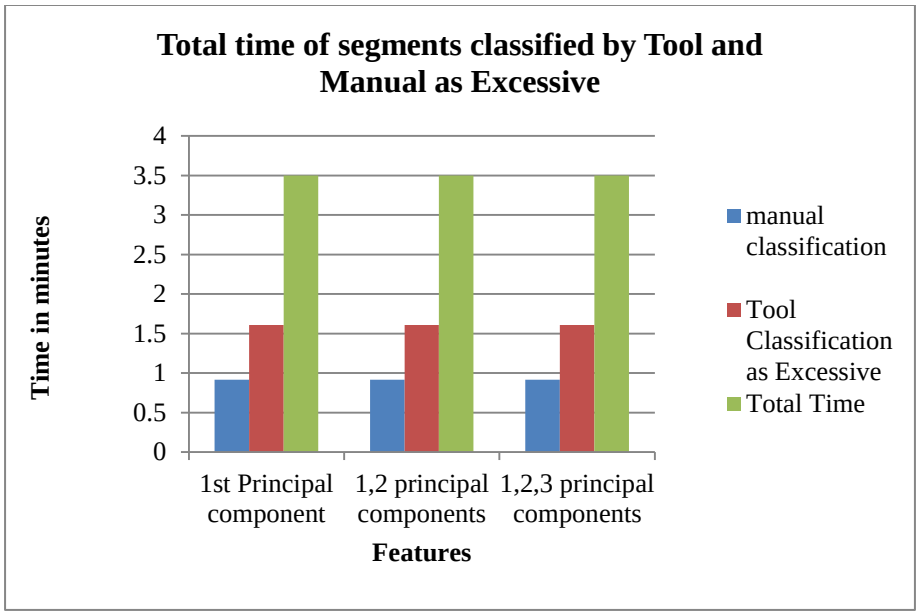

Figure 40 - Total time of segments classified as excessive by the software program and manually

The green bars in Figure 40 represent the total time of video recorded by the eye tracker. Manual classification of the video file, depicted by the blue bars, shows 0.91 minutes of excessive effort. The automatic classification of the video file for all the three features shows 1.61 minutes of excessive effort time which is depicted by the maroon bars in figure 40. Even though the total time of segments classified as excessive is less, the percentage of type-II errors is very high and not within the acceptable limit for all the features.

Data file 5 Results:

The video file corresponding to data file 3 is 5.65 minutes in length. Figure 41 shows the percentage of E vs. E, E vs. NE, NE vs. NE and NE vs. E segments for the features mentioned in the experiment description.

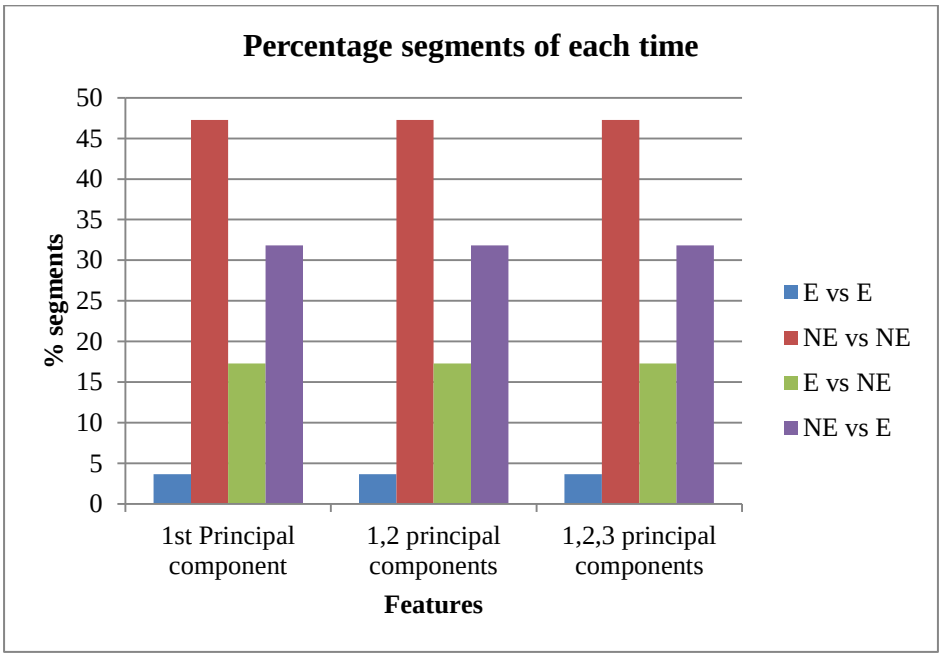

Figure 41 - Graph of percentage of segments of each type

From the graph in figure 41 it is clear that all the three features have same percentage of E vs. E, E vs. NE, NE vs. NE and NE vs. E segments.

Figure 42 shows the total time of segments classified as excessive by the software program and the manual process for the three features.

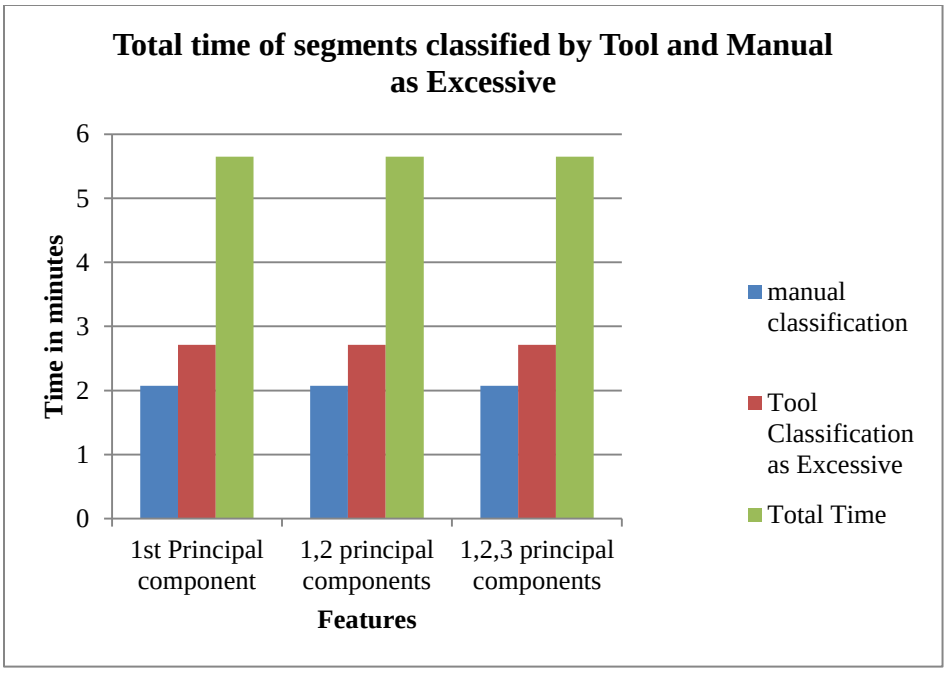

Figure 42 - Total time of segments classified as excessive by the software program and manually

The green bars in figure 42 represent the total time of video recorded by the eye tracker. Manual classification of the video file, depicted by the blue bars, shows 2.07 minutes of excessive effort. The automatic classification of the video file for all the three features shows 2.71 minutes of excessive effort time which is depicted by the maroon bars in figure 42. Even though the total time of segments classified as excessive is less, the percentage of type-II errors is very high and not within the acceptable limit for all the features.

CHAPTER 6

RESULT ANALYSIS

6.1 Evaluation

In this chapter, I evaluate and discuss the results of the experiments conducted in this work. My criteria for success are based on 1) The number of type-II errors and 2) A minimal time to investigate usability issues with an acceptable level of type-II errors. My assumption is that 15% of error of type-II is the upper bound for being considered as acceptable. The results are evaluated based on the performance of each pattern recognition method on individual features. In addition, the overall performance of each pattern recognition method is evaluated.

To recapitulate, segments that are classified as non-excessive effort segments by the manual classification process but classified as excessive effort segments by the software program are classified as type-I errors. Contrarily, type-II errors depict those segments identified as non-excessive by the software program but are actually excessive effort per manual classification. In the case of type-I errors, the software program is highlighting extra segments that need further investigation. The total inspection time is the sum of the duration of each segment classified as excessive by the software program.

The index of items listed in tables 1 through 8 is:

- # Fix - denotes number of fixations

- Avg Fix Dur - denotes average fixation duration
- # Sacc - denotes number of saccades
- Sacc Amp - denotes average saccade amplitude
- Eye Path - denotes eye path traversed
- FPC - denotes first principal component

Table 1, shows the results of experiment 1 conducted on the five data files. The results are grouped based on the data file.

Table 1 - Experiment 1 Summary

Exp #	Data file	Feature value	% type -I errors	% type -II errors	% of total errors	Total video time	Inspection time	Inspection time as a % of total time
1	1	# Fix	20.43	9.677	30.108	6.09	3.0132833	49.479201
1	1	Avg Fix Dur	41.935	15.05	56.989	6.09	2.1342333	35.044882
1	1	#Sacc	26.882	10.75	37.634	6.09	3.39655	55.772578
1	1	Sacc Amp	37.634	12.9	50.538	6.09	2.5303667	41.549535
1	1	Eye Path	26.882	11.83	38.71	6.09	3.1481	51.692939
1	2	# Fix	45.745	3.191	48.936	3.27	1.87315	57.282875
1	2	Avg Fix Dur	40.426	5.319	45.745	3.27	1.05115	32.14526
1	2	#Sacc	34.043	4.255	38.298	3.27	1.7260667	52.784913
1	2	Sacc Amp	27.66	4.255	31.915	3.27	1.3571333	41.502548
1	2	Eye Path	26.596	5.319	31.915	3.27	1.3856	42.373089
1	3	# Fix	2.0962	0.041	2.1375	3.802	2.4455833	64.323602
1	3	Avg Fix Dur	2.2465	0.702	2.948	3.802	2.3267333	61.197615
1	3	#Sacc	2.5022	0.01	2.5122	3.802	2.6001167	68.388129
1	3	Sacc Amp	1.594	0.191	1.7854	3.802	2.3243833	61.135806
1	3	Eye Path	1.1861	0.031	1.2167	3.802	2.07625	54.609416
1	4	# Fix	49.333	0	49.333	3.5	2.6636167	76.103333
1	4	Avg Fix Dur	20	12	32	3.5	0.7377	21.077143
1	4	#Sacc	45.333	0	45.333	3.5	2.63455	75.272857
1	4	Sacc Amp	49.333	1.333	50.667	3.5	2.6396333	75.418095
1	4	Eye Path	44	2.667	46.667	3.5	2.5187167	71.963333
1	5	# Fix	24.545	3.636	28.182	5.65	3.5643333	63.085546
1	5	Avg Fix Dur	42.727	16.36	59.091	5.65	2.1119833	37.380236
1	5	#Sacc	0	37.27	37.273	5.65	3.8475	68.097345
1	5	Sacc Amp	29.091	4.545	33.636	5.65	3.5159167	62.228614
1	5	Eye Path	30	5.455	35.455	5.65	3.8452833	68.058112

From table 1 it is observed that in most of the data files, the number of fixations performs well in terms of minimum type-II errors, while not introducing high inspection time. In all the data files, the average fixation duration has the best inspection times, but

also the highest percentage of type-II errors. Table 2, shows the average value of all the features used in experiment 1 over five data files.

Table 2 - Average values of experiment 1 results

Exp #	Feature value	avg. # of excessive effort segments	avg. total no of segments	avg. % type- I errors	avg. % type- II errors	avg. % of total errors	avg. Inspection time	avg. Inspection time as a % of total time
1	# Fix	17.2	95	28.429962	3.3092978	31.73926	2.7119933	62.054911
1	Avg Fix Dur	18.2	95	29.466961	9.8876097	39.354571	1.67236	37.369027
1	#Sacc	32	95	21.751965	10.458144	32.210108	2.8409567	64.063165
1	Sacc Amp	17.6	95	29.062445	4.6457432	33.708188	2.4734867	56.36692
1	Eye Path	17.8	95	25.732703	5.0597969	30.7925	2.59479	57.739378

The following are the observations derived from table 2.

- The results of table 2 show that the threshold method on the feature value, *number of fixations*," gives good results in terms of type-II errors but, the average inspection time is relatively high when compared to other feature values. The average value of type-II errors for number of fixations is 3.3%. Average saccade amplitude and eye path traversed follow number of fixations in terms of type-II errors.
- A threshold on *average fixation duration* performs well in terms of minimal inspection time with an acceptable value of 9.8% for type-II errors.
- A feature value with minimum number of total errors is *eye path traversed*. This feature value is a good choice when inspection time is not taken into account.

- The inspection time is not completely correlated to type-I errors. In the case of average fixation duration, the inspection time is 1.67 minutes with 29.4% of type-I errors. On the other hand, the average saccade amplitude with almost the same percentage of type-I errors has higher inspection time than average fixation duration.
- The values of the average number of excessive effort segments for all features are in close proximity to each other. However, the percentage of type-I and type-II errors differ invariably. This portrays that the segments classified as excessive are different for each feature value.
- Despite the fact that the percentages of total errors for each feature value is in close proximity to each other, the inspection times vary. This delineates that the segments classified as excessive are different for each feature value.

Table 3, shows the results of experiment 2 conducted on five data files. The results are grouped based on the data file.

Table 3 - Experiment 2 Summary

Exp #	Data file	Feature value	% type -I errors	% type -II errors	% of total errors	Total video time	Inspection time	Inspection time as a % of total time
2	1	#fix	9.6774194	13.978495	23.655914	6.09	2.2734167	37.330323
2	1	#sacc	9.6774194	13.978495	23.655914	6.09	2.2734167	37.330323
2	1	eye path	19.354839	15.053763	34.408602	6.09	2.5492	41.858785
2	1	#fix, #sacc, eye path	8.6021505	13.978495	22.580645	6.09	2.0993833	34.472633
2	1	#fix, #sacc, eye path, avg fix dur, avg sacc amp	58.064516	1.0752688	59.139785	6.09	4.3279833	71.06705
2	2	#fix	45.744681	4.2553191	50	3.27	1.87315	57.282875
2	2	#sacc	24.468085	5.3191489	29.787234	3.27	1.49485	45.714067
2	2	eye path	14.893617	8.5106383	23.404255	3.27	0.8030167	24.557085
2	2	#fix, #sacc, eye path	22.340426	6.3829787	28.723404	3.27	1.2937167	39.563201
2	2	#fix, #sacc, eye path, avg fix dur, avg sacc amp	24.468085	7.4468085	31.914894	3.27	1.3016833	39.80683
2	3	#fix	41.904762	1.9047619	43.809524	3.802	2.9758833	78.271524
2	3	#sacc	22.857143	3.8095238	26.666667	3.802	2.0379333	53.601613
2	3	eye path	12.380952	7.6190476	20	3.802	1.1789333	31.008241
2	3	#fix, #sacc, eye path	20	4.7619048	24.761905	3.802	1.7791833	46.795985
2	3	#fix, #sacc, eye path, avg fix dur, avg sacc amp	40	1.9047619	41.904762	3.802	2.6101167	68.651149
2	4	#fix	30.666667	2.1052632	32.77193	3.5	2.32	66.285714
2	4	#sacc	29.333333	6.6666667	36	3.5	1.98	56.571429
2	4	eye path	25.333333	9.3333333	34.666667	3.5	1.84	52.571429
2	4	#fix, #sacc, eye path	36	5.3333333	41.333333	3.5	2.29	65.428571
2	4	#fix, #sacc, eye path, avg fix dur, avg sacc amp	36	4	40	3.5	2.31	66
2	5	#fix	8.1818182	10.909091	19.090909	5.65	2.35	41.59292
2	5	#sacc	2.7272727	14.545455	17.272727	5.65	1.82	32.212389
2	5	eye path	0.9090909	14.545455	15.454545	5.65	1.42	25.132743
2	5	#fix, #sacc, eye path	4.5454545	12.727273	17.272727	5.65	2.04	36.106195
2	5	#fix, #sacc, eye path, avg fix dur, avg sacc amp	4.5454545	12.727273	17.272727	5.65	2.04	36.106195

From table 3 it is observed that the feature subset with the features - number of saccades, eye path traversed, average fixation duration, average saccade amplitude, and number of fixations performs well in terms of minimum type-II errors in data files 1 and 3. However, the number of fixations performs well in data files 2, 3, 4 and 5 in terms of type-II errors while not putting too much focus on the inspection time. In all the data files with an exception of data file 1, eye path traversed has the best inspection times, but also the highest percentage of type-II errors. It is also evident that the percentages of type-I and type-II errors vary by a wide range between each feature subset for any given data file. Table 4, shows the average value of all the features used in experiment 2 over five data files.

Table 4 - Average values of experiment 2 results

Exp #	Feature value	avg. # of excessive effort segments	avg. total no of segments	avg. % type -I errors	avg. % type -II errors	avg. % of total errors	avg. Inspection time	avg. Inspection time as a % of total time
2	#fix	29.076336	95	27.235069	6.6305859	33.865655	2.35849	56.152671
2	#sacc	23.542982	95	17.812651	8.8638577	26.676508	1.92124	45.085964
2	eye path	19.749442	95	17.990685	10.129196	28.119881	1.5927875	37.498885
2	#fix, #sacc, eye path	23.236658	95	18.297606	8.6367968	26.934403	1.9004567	44.473317
2	#fix, #sacc, eye path, avg fix dur,avg sacc amp	29.163122	95	32.615611	5.4308224	38.046434	2.5179567	56.326245

The following are the observations derived from the above table:

- The results from table 4 show that the K-means clustering on the *feature subset- number of fixations, number of saccades, eye path traversed, average fixation duration, and average saccade amplitude,* gives good results in terms of type-II

errors with an average value of 5.4%. But, the average inspection time is relatively high when compared to other feature values. Number of fixations follow the above identified feature value in terms of type-II errors.

- Clustering on the *eye path traversed* performs well in terms of minimal inspection time with an acceptable value of 10.1% for type-II errors.
- A feature value with minimum number of total errors is number of fixations. This feature value is a good choice when inspection time is not taken into account.
- The average number of excessive effort segments for number of fixations and the feature subset with the following features - number of saccades, eye path traversed, average fixation duration, average saccade amplitude are the same. However, the inspection times vary. This portrays that the segments classified as excessive are different for each feature value.
- Unlike the results of the threshold method, the percentages of total errors for each feature value vary by a wide margin when applying the K-means clustering on the feature subsets.

Next, the results of applying the threshold method on the first principal component are summarized in Table 5.

Table 5 - Experiment 3 summary

Exp #	Data file	Feature value	% type -I errors	% type- II errors	% of total errors	Total video time	Inspection time	Inspection time as a % of total time
3	1	FPC	26.88172	9.6774194	36.55914	6.09	3.3227667	54.561029
3	2	FPC	29.787234	4.2553191	34.042553	3.27	1.5256167	46.654944
3	3	FPC	1.39155	0.0306667	1.4222167	3.802	2.28175	60.014466
3	4	FPC	46.666667	1.3333333	48	3.5	2.6298667	75.139048
3	5	FPC	32.727273	5.4545455	38.181818	5.65	3.9240333	69.451917

Data file 2 and data file 3 present the most interesting observation from the above table. In case of data file 2, the total errors are relatively very high but the inspection time is small and the converse is true for data file 3. Total time of segments classified as excessive play a significant role in determining inspection time rather than type-I and type-II errors. The type-II errors are all relatively low and are within an acceptable limit. Table 6, shows the average values of experiment 3.

Table 6 - Average values of experiment 3

Exp #	Feature value	avg. # of excessive effort segments	avg. total no of segments	avg. % type -I errors	avg. % type- II errors	avg. % of total errors	avg. Inspection time	avg. Inspection time as a % of total time
3	1st principal components	16.6	95	27.490889	4.1502568	31.641146	2.7368067	61.164281

The results of this experiment are compared with the results obtained from Experiment 1 to analyze the performance of the threshold method on first principal component with other features like: 1) number of fixations, 2) average fixation duration, 3) number of saccades, 4) average saccade amplitude, and 5) eye path travelled.

Experiment 1 result evaluation shows that the feature value, number of fixations, gives good results in terms of type-II errors. The average percentage of type-II errors for number of fixations is 3.3%, whereas it is 4.1% for first principal component. Initially, average saccade amplitude and eye path traversed succeeded number of fixations in terms of performance. However, the new results place a threshold on the *first principal component* after the number of fixations with respect to type-II errors. The inspection time for first principal component and average fixation duration are 2.7 and 1.6 minutes respectively. A threshold on *average fixation duration* performs better than first principal component in terms of lower inspection time and an acceptable 9.8% for type-II errors.

Table 7 summarizes the results from experiment 4. Applying the K-means clustering on different combinations of the principal components yield the same results for type-I, type-II and inspection time. In this table, the values on each row are the results for the three features – 1st principal component, 1st and 2nd principal component, and 1st, 2nd and 3rd principal component.

Table 7 - Experiment 4 summary

Exp #	Data file	Feature value	% type -I errors	% type-II errors	% of total errors	Total video time	Inspection time	Inspection time as a % of total time
4	1	1st, 2nd & 3rd principal components	24.731183	17.204301	24.731183	6.09	2.5	41.050903
4	2	1st, 2nd & 3rd principal components	26.88172	5.3763441	32.258065	3.27	1.03	31.498471
4	3	1st, 2nd & 3rd principal components	20	8.5714286	28.571429	3.802	1.95	51.288795
4	4	1st, 2nd & 3rd principal components	18.666667	14.666667	33.333333	3.5	1.61	46
4	5	1st, 2nd & 3rd principal components	31.818182	17.272727	49.090909	5.65	2.71	47.964602

By applying the K-means clustering on different combinations of the principal components, it is observed that the inspection times are relatively low. However, the type -II error percentages are high. Only data file 2 and data file 3, results show low values for type-II errors.

Table 8 - Average values of experiment 4

Exp #	Feature value	avg. # of excessive effort segments	avg. total no of segments	avg. % type- I errors	avg. % type- II errors	avg. % of total errors	avg. Inspection time	avg. Inspection time as a % of total time
4	1st, 2nd & 3rd principal components	28.6	95	24.41955	12.618294	37.037844	1.96	43.560554

Table 8, shows the average values of all the features used in experiment 4 over five data files. The average type-II error is very high when using the K-means on the principal components. The average inspection time is only 1.96%. When taking type-II errors also into consideration, this method is not suitable to identify excessive effort segments.

Of all the pattern recognition methods used, a threshold on number of fixations yields the best results in terms of type-II errors and is followed by a threshold on first principal component. K-means clustering on feature subset with the features: number of fixations, number of saccades, average saccade amplitude, average fixation duration, and eye path traversed ranks third.

When time to investigate is taken into consideration while also confirming that the type-II errors are within a reasonable limit, the K-means clustering on the number of

saccades yields the best results and precedes the threshold method on average fixation duration in performance.

CHAPTER 7

CONCLUSION AND FUTURE RESEARCH

7.1 Conclusions

The framework presented in this research enables software developers to efficiently identify usability issues thereby optimizing the time spent on usability testing. Excessive effort segments, which typically relate to usability issues, are identified by applying pattern recognition techniques such as K-means clustering algorithm, thresholding, principal component analysis, and feature selection. The analysis of the experiments conducted in this paper show that the time taken for usability testing can be reduced by 40% or more.

7.2 Recommendations for Future Research

With greater time and resources at one's disposal, there is scope to enhance the definition and implementation of pattern recognition techniques in identifying usability issues in software.

In this research, the time between two consecutive keyboard/mouse clicks by user is considered as a segment and this has served as the basic pattern for pattern recognition techniques. Equal time slicing of user's software interaction session can be used instead and the performance results can be analyzed and compared with the results from this research.

Further refinement of pattern recognition techniques can be pursued to minimize errors and inspection time. Also, more focus can be given to the criteria for manual classification of video segments thus allowing excessive effort segments to be identified more accurately.

Another direction for future research is to automate some of the manual steps in this process. This can include software that automatically log users' software interaction session data, manipulate data, and without human intervention lists the start and end times of all the excessive effort segments. This can drastically reduce time taken for usability testing.

In this work, I have concentrated on pattern recognition techniques that do not rely on human intelligence. Hence, results are generated using non-supervised learning procedures. A surrogate approach can be to use supervised learning procedures to produce the output. This involves conducting experiments using training data sets to manually arrive at an archetype that can be applied on any data set to generate the output.

Another direction for further research is to consider information fusion. Information fusion combines different techniques of pattern recognition classification to achieve more accurate results. Another approach that can be researched is to arrive at a formula or function that can be applied to any data set by which usability deficiencies can be identified.

BIBLOGRAPHY

[1] Dumas, J S, and J C Redish. "A Practical Guide to Usability Testing." *Intellect Books.* Portland, OR, USA, 1999.

[2] Nielsen, J. "Usability Engineering." San Francisco, CA, USA: Academic Press, 1993.

[3] Pressman, R. *Software Engineering: A Practitioner's Approach.* New York, NY: McGraw-Hill, 2005.

[4] Rubin , J, and D Chisnell. *Handbook of Usability Testing: How to Plan , Design, and Conduct Effective Tests.* Indianapolis, IN, USA: Wiley Publishing, Inc., 2008.

[5] Tullis, T, and B Albert. *Measuring The User Experience: collecting, analyzing, and presenting usability metrics.* Burlington, MA: Morgan Kaufmann, 2008.

[6] Mueller, C, D Tamir, O Komogortsev, and L Feldman. "Using Designer's Effort for User Interface Evaluation." *2009 IEEE International Conference on Systems, Man, and Cybernetics.* San Antonio, Texas, USA, October 11, 2009.

[7] Mueller, Carl, Dan Tamir, Oleg Komogortsev, and L Feldman. "An Economical Approach to Usability Testing." *Proceedings of the 33rd Annual IEEE International Computer Software and Applications Conference.* Seattle, Washington, July 2009.

[8] ISO 9241-11. "Ergonomic requirements for office work with visual display terminals (VDTs) - Part 11: Guidance on usability." Geneva, Switzerland: International Standards Organization, 1998.

[9] ISO/IEC 9126-1:2001 . "Software Engineering-Product Quality-Part 1: Quality Model." Geneva, Switzerland: International Standards Organization, 2001.

[10] Komogortsev , O, C Mueller, D Tamir, and L Fledman. "An Effort Based Model of Software Usability." *2009 International Conference on Software Engineering Theory and Practice.* Orlando, FL, 2009.

[11] Tou, J T, and R C Gonzalez. *Pattern Recognition Principles.* USA: Addison-Wesley Publishing, Inc., 1974.

[12] ISO/IEC 9126-1:2001. "Software Engineering-Product Quality-Part 2: External Metrics." Geneva, Switzerland: International Standards Organization, 2001.

[13] Tamir, D E, O V Komogortsev, and C J Mueller. "An Effort and Time Based Measure of Usability." *6th Workshop on Software Quality,30th International Conference on Software Engineering.* Leipzig, Germany, May 2008.

[14] Poole , A, and L J Ball. *Eye Tracking in Human-Computer Interaction and Usability Research: Current Status and Future Prospects.* Encyclopedia of Human Computer Interaction: Idea Group, 2004.

[15] Just, M A, and P A Carpenter. "Eye Fixation and Cognitive Processes." Cognitive Psychology 8: 441-480, 1976.

[16] Ebbinghaus, H. *Memory: A Contribution to Experimental Psychology.* 1885.

[17] Tamir, Dan, and Carl Mueller. "Pinpointing Usability Issues Using an Effort Based Framework." *to appear in the IEEE Systems, Man, Cybernetics Conference.* Istanbul, Turkey, 2010.

[18] Tamir, D E, Divya Dasari Kali Venkata, et al. "Detection of Software Usability Deficiencies." *HCI*, 2011.

[19] Tamir, D E, and A Kandel. "The Pyramid Fuzzy C-means Algorithm." *International Journal of Computational Intelligence in Control, 2 (2)*, 2012.

[20] Duda, R O, P E Hart, and D G Stock. *Pattern Classification, 2nd E.* Willey International, 2001.

[21] Ebba, Thora Hvannberg, and Chong Law Lai. "Classification of Usability Problems (CUP) Scheme." *Nordic conference on Human-computer Interaction.* 2006.

[22] Noboru, Nakamichi, Sakai Makoto, and Shim Kazuyuki . "Detecting Low Usability Web Pages using Quantitative Data of Users' Behavior." *IEEE.* 2009.

[23] Makoto, Sakai, Nakamichi Noboru, Hu Jian, Shima Kazuyuki, and Nakamura Masahide. "Webtracer: A New Integrated Environment for Web Usability Testing ." *10th Int'l Conference on Human - Computer Interaction.* Crete, Greece: HCI International, June 2003. 289-290.

[24] Anonymous, "MATLAB Product Help," MATLAB 7.10.0.99

VITA

Dasari Kali Venkata Divya, the daughter of Dasari Visweswara Rao and Dasari Pushpa, was born on December 2, 1985 in Andhra Pradesh, India,. She is the wife of Lenin Sana and they are expecting their first child in March, 2013. She received a Bachelor of Engineering in Electrical and Electronics Engineering from Anna University, India in May 2007. In August 2009, she entered the Graduate College of Texas State University-San Marcos.

Permanent Email: dkvdivya@gmail.com

This thesis was typed by Dasari Kali Venkata Divya.