An Identification System for Head Mounted Displays

A major Qualifying Project Report:

Submitted to the Faculty

Of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

By:

_____

Cynthia Rogers

_____

Alexander Witt

_____

Alexander Solomon

Date: March 10, 2015

Approved:

_____

Prof. Krishna Venkatasubramanian, Advisor

Contents

List of Figures

List of Tables

Abstract

Personalized devices often require a form of user identification to provide customized performance and rudimentary privacy between a limited number of users. Because of the personal nature of head mounted devices, the new and growing industry of head mounted displays requires a method to identify users to increase the customizability and usability of such devices. This project introduces a system that accurately identifies users with common sensors included on head mounted displays. The proposed system records user blink behavior, head position and head movement and then uses high dimensional machine learning algorithms to identify users based on trends in their collected data. The system demonstrated over 98% accuracy, demonstrating its ability to identify users.

1 Introduction

Personalized devices often require a form of user identification to provide customized performance and rudimentary privacy between a limited number of users. Because of the personal nature of **head mounted devices** (HMDs), the new and growing industry of HMDs requires a method to identify users in order to increase the customizability and usability of such devices. Examples of some HMDs that have received public attention include Sony's Project Morpheus [14], Facebook's Oculus Rift [19], Microsoft's Hololens [16], and Google Glass. Because many of these existing HMDs have limited input vectors, traditional identification methods become cumbersome or impossible. Inapplicable traditional identification methods typically include entering a textual password or input combination. Not only are some HMDs incapable of accepting textual input, but many HMDs are designed for hands-free usage. A hands-free identification system thus becomes desirable. One common method for implementing hands-free identification is to use biometrics.

A **biometric** is a physical characteristic or behavior that can uniquely identify an individual. Commonly referenced biometrics include fingerprints, retinal scans and voice recognition. While some biometrics are more unique than others, combining biometrics often allows for a system to be more accurate when performing the task of identification [7]. Consequently, it is beneficial for HMDs to use as many input sensors as possible to identify a user in a hands-free manner. This report introduces a software-based system that accurately identifies users through the use of common sensors included on head mounted displays.

In particular, the developed system relies on data collected from a gyroscope to measure head movement, data collected from an accelerometer to measure head position and data collected from an IR sensor to measure blink behavior. This information is collected from the user through an application that alternatively displays a static image to a user and then asks the user to locate specific images that are flashed in rapid sequence. Once this information is recorded, the system then uses high dimensional **machine learning** (ML) algorithms to perform identification based on trends in the user's collected data. Different sub-sets of the data (i.e. blink and head movement data) were then compared to the full data set in order to determine which types of recorded information were more capable for supporting identification.

Analysis of the system from a group of 13 test subjects demonstrated high performance when applying the most successful ML algorithm, Functional Tree, across the full data set. The Functional Tree algorithm demonstrated a 98% accuracy with a false positive and false negative rate lower than 16%. The high accuracy rates and low false positive and false negative rates reported for the ML algorithms effectively demonstrate that the proposed system was successful at performing the task of identification across the group of 13 users.

This report is organized so that the second section describes the problem statement. The third section discusses work related to the research that was conducted. The fourth section provides the background information necessary for understanding the remainder of the presented work. The fifth section describes the design of the system. The sixth section describes system implementation specifics such as platform choice and software design. The seventh section describes the system performance analysis. The eighth section provides a discussion of the developed system and the performance analysis. The ninth section concludes the report.

2 Problem Statement

HMDs are steadily becoming more prominent. With many companies investing in HMDs, the need for user identification on these devices is a growing expectation. A system that demonstrates user identification is a system capable of differentiating between the individuals who use that system. A collection of reliable biometrics that are capable of being used with head-mounted technology must be established in order to successfully implement identification. HMDs that have the ability to collect data on blinking behavior have a high potential for using blink-based biometric identification. Because many HMDs already tend to record user head movement and head position, our research includes the use of head movement behavioral biometrics as well.

The objective of the performed research is to determine whether biometric data such as an individual's blinking and head movement could be collected with existing HMDs and reliably used for the purpose of identifying different users in a system.

3 Related Work

The related work of our paper primarily focuses on both the topic of biometrics and biometrics that are based on observed blinking behavior in humans.

The system that we developed relies on the ability to collect blink and head movement data from human subjects. Previous research in which blinking behavior was detected and analyzed in order to perform authentication has been performed with voluntary blinking to song cadences [24]. Our research differs from blink-based biometrics that are deduced from song by permitting users to blink naturally while viewing a sequence of images and attempting to locate specific images in that sequence. Additionally, another key difference between our work and the system developed for use with song cadences was the size of our user set. The song cadence based system required a high degree of precision and had a user set of only four individuals while our proposed system has a user set of thirteen individuals. Finally, while the system developed for use with song cadences did demonstrate the ability to utilize blinking behavior as a biometric, our research expands upon this idea by eliminating the need for song cadences and applying biometrics based on blinking to HMDs.

Apart from analyzing blinking in a conventional way, research using wearable technology to detect blinking behavior via **an infrared sensor** (IR sensor) has been conducted in order to identify the actions being performed by individuals [9]. While this research was useful for demonstrating how to detect blinking behavior while using an HMD, it deviated from our research objective because its primary concern was to identify types of actions and not individuals. Additionally, our prototype system was designed with the intent to collect the natural blinking and head movement data of a user from an HMD in order to identify a user who is sitting in a still position.

Head movement detection was collected as an additional type of data for our system. Due to slight head movements and posture often being associated with an individual, head movement data was considered to be valuable because it would provide more integrity to a blink-based biometric [7].

The collection of related work above was not sufficient to demonstrate a system for user identification on an HMD because the research was either not performed on an HMD or its main purpose was not the identification of users on the system. Our goal is to develop a system capable of differentiating between individuals who use an HMD.

4 Background

This section describes the different types of data collected in the system. In particular, we collect blink and movement type data. To define blink data we discuss our concept of a blink and communicate the varying blink types we reference in this report. To define movement data, we discuss head movement orientation and head movement types referenced in the report.

4.1 Blink Mechanics

Blinking is a mechanical function found to occur in most humans, and its frequency of occurrence is generally known to vary both for an individual and between individuals. Overall, there are two general types of blinking, and they are voluntary blinking and involuntary blinking. Voluntary blinking occurs when an individual wills themselves to blink and involuntary blinking happens naturally and without thought. In addition, there are also two common lid-contact-based classifications for blinking; a blink may either be complete or incomplete. When a blink is complete, the two lid margins of the human eye come in contact with one another; however, when a blink is incomplete, the two lid margins of the human eye do not come in contact with one another.

With respect to the eye, the main physiological reason for the mechanical function of blinking pertains to variations in the ocular surface. In particular, when the tear-film that coats the cornea of the eye begins to degrade, blinking provides a means of replenishing that tear-film [1]. The process of blinking performs this task by stimulating the secretion of glands such as *mucin* and *meibum* over the surface of the eye [1]. Additional means by which the tear film can be disturbed typically involve characteristics of the environment. For instance, an increase in the amount airborne particulate matter such as pollen, dust, pollutants, etc… can often result in an observed increase in blink rate [1]. Other aspects of the environment, such as humidity and ambient temperature can also influence the integrity of the tear-film through their relationship to its evaporation [10].

Ocular surface conditions alone do not fully account for all instances of observed blinking in humans. Additional factors that have been known to significantly influence the mechanics of blinking are cognitive state, emotional state, and visual fixation [1]. In particular, cognitive state has been shown to increase the blinking rate in test subjects when tasks of increasing mental load are performed [1][3]. A study had indicated that in a test of 150 normal subjects under relatively constant conditions, 67.3% of participants had exhibited a trend where the rate of blinking during conversation was greater than the rate of blinking during inactivity which in turn was greater than the rate of blinking while reading [3]. 98.7% of participants were shown to exhibit greater blinking frequency during conversation than during reading [3]. These results effectively indicate that cognitive state does significantly influence blinking.

In regard to visual fixation, a study observed a decrease in blink rate while normal test subjects viewed electronic video content [26]. In particular, the findings of that study had shown an average 500% decrease in blink rate while watching content shown on a standard CRT television screen; the mean blink rate of subjects prior to watching a standard CRT television screen was 18.4 blinks/min (SD = ±5.7)

and the mean blink rate of subjects while watching was 3.6 blinks/min (SD = ±1.8) [26]. Therefore it is evident that the rate of blinking is dramatically reduced when subjects focus directly on video content.

It is important to note, however, that this reduction in the rate of blinking is not the only effect that video displays units can have on the mechanics of blinking. As some studies have shown, content presented through a video display unit can potentially induce the occurrence of blinks during explicit and implicit shifts in the content being relayed [23]. In particular, a study indicated that viewing a stream of related visual events can result in synchronized patterns of blinking with minor latency differences among test subjects [23]. Whether or not this synchronization remains in effect during tasks that do not present a logically flowing stream of visual events remains uncertain. Consequently, in addition to the factors that influence blinking (see Table 1), there are essentially two different forms of blinking behavior that can be monitored during the display of visual content; blinks that occur naturally (i.e. **spontaneous blinks**) which we will denote as **SB** in this report and blinks that can result from exposure to an active stream of visual content (i.e. **induced blinks**) which we will denote as **IB** in this report.

Additionally, the environmental and cognitive influencers of blinking may have potential application as a means of associating individuals with the patterns of blinking that they tend to exhibit under different conditions. See Table 1 in order to view the factors that have been shown to influence blink rate [3][4][9][10][11][12][17].

Table 1: Effect of cognitive and environmental factors on blinking

Factors	Increases Blink Rate	Decreases Blink Rate
Airborne Particulate Matter	✓	
Increased Humidity		✓
Visual Fixation		✓
Longer Maximum Blink Interval	✓	
Ocular Surface Damage	✓	

Factors	Increases Blink Rate	Decreases Blink Rate
Parkinsonism		✓
Schizophrenia	✓	
Fatigue		✓
Exposed Ocular Surface Area	✓	
Speech or Memory Tasks	✓	
Increased Cognitive Task Difficulty	✓	
Increased Time on Task		✓
High Level of Illumination	✓	
Low Level of Illumination	✓	

4.2 Movement Data

In addition to being able to record blink-based biometric data, head movement data is another biometric that is capable of being utilized. Because head movement is intuitively known to be influenced by both posture and slight anatomical differences between individuals such as in the physiology of neck muscles [2], head movement provides a potential source of biological uniqueness among individuals. Additionally, head movement provides a wealth of data because, in order to successfully monitor head movement, both the direction and magnitude of a head movement must be considered. In order to model the direction of a given head movement, the aviation terminology of pitch, yaw, and roll are often used. A pitch in relation to head movement is when the head nods (in western cultures). A yaw relates to the axis that a person shakes their head on (in western cultures). A roll is when the head leans to either side (see Figure 1).

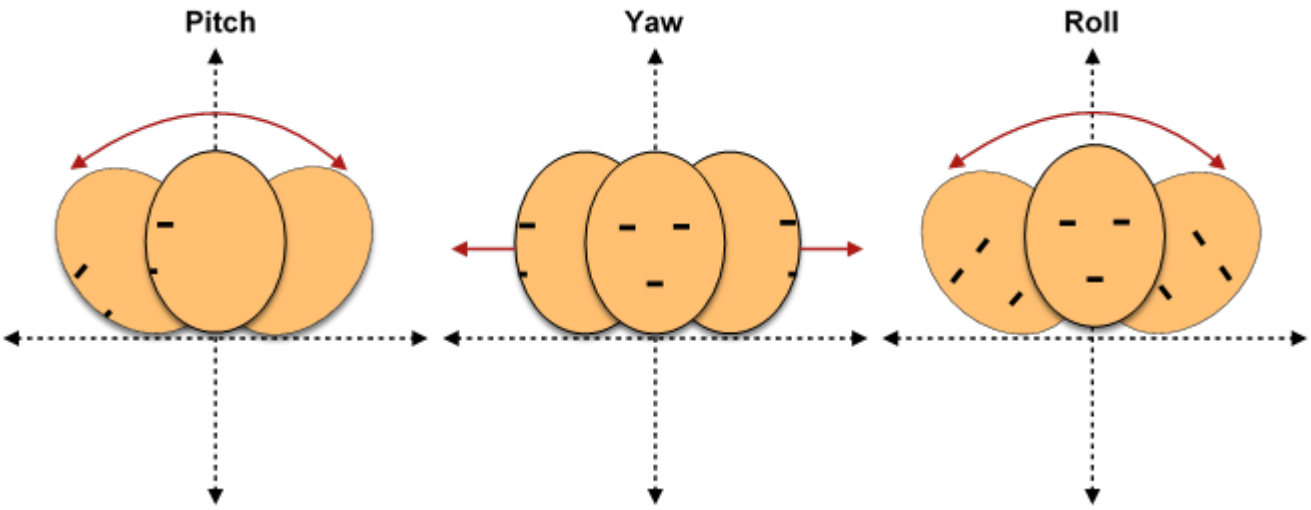

Figure 1: Pitch, yaw, and roll in terms of head movement

Similar to how head movements are classified from the standpoint of directionality, monitoring the magnitude of head movements also occurs in three dimensions. This is specifically done by monitoring magnitude readings for a given head movement in the x, y, and z-axes of a Cartesian coordinate system (see Figure 2).

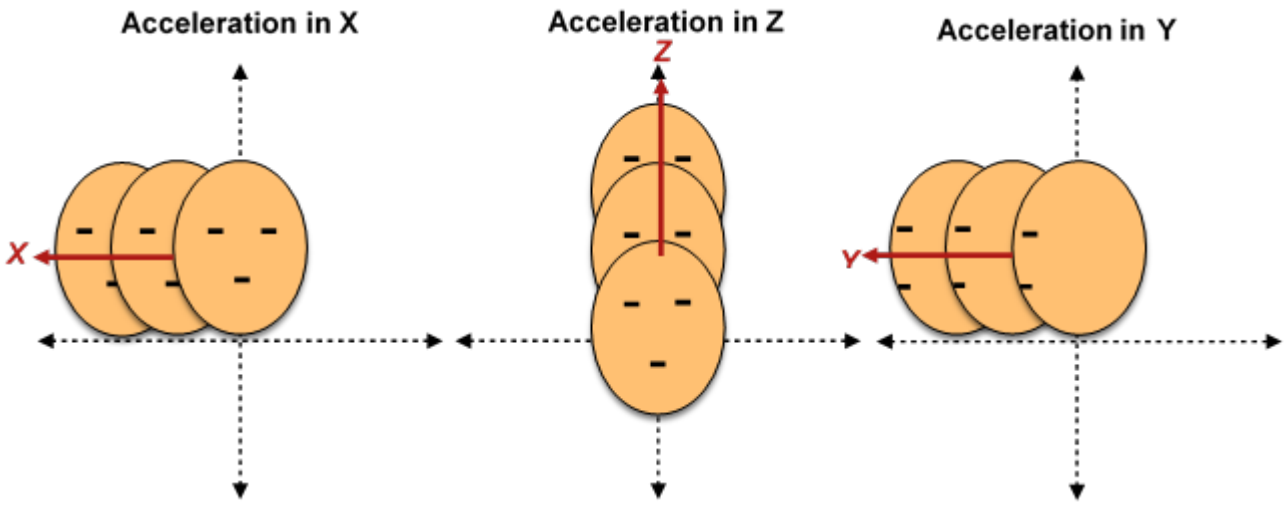

Figure 2: Acceleration directions in terms of head movement

5 System Design

Viewed from a high-level of abstraction, the system designed for this paper was comprised of three operational components. These components consisted of a stage for data collection, a stage for processing data, and a stage for training an ML classifier. When combined in the sequential order displayed in Figure 3, the result was a software-based system capable of retrieving, processing, and classifying recorded blink and head movement data.

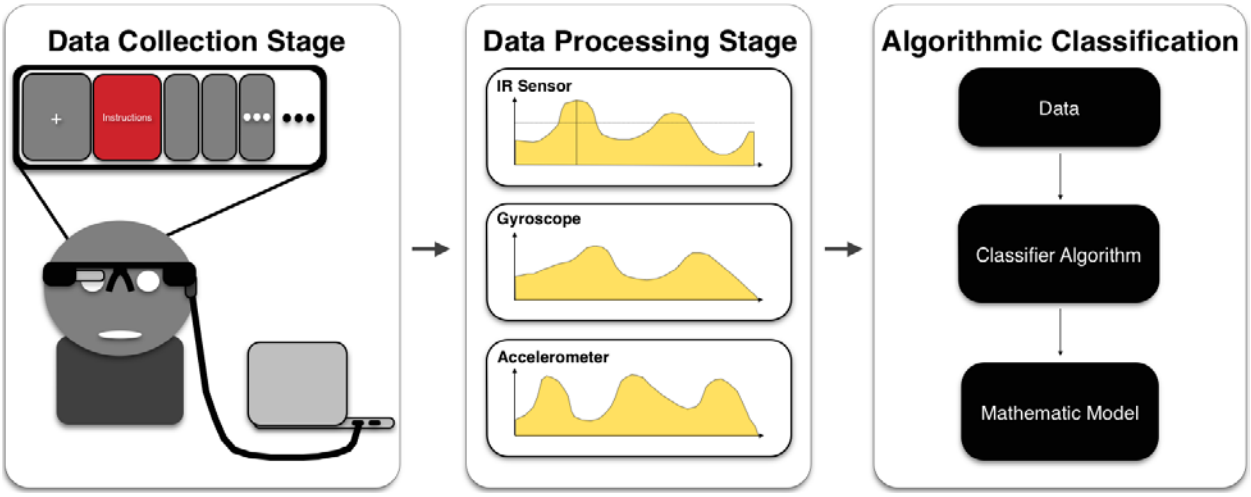

Figure 3: System design of data collection to data processing to ML analysis

To create a system, which used blink and head movement based biometrics to identify users we first collected raw data from an HMD. We extract features from this raw data, which describe the data in terms of quantified biometric vectors. Once features are extracted from the data, we use different ML algorithms to predict a user's identity based on the features which describe their raw data.

5.1 Data Collection Stage

When using the system during the data collection stage, the three main sensors that we poll are an infrared (IR) sensor, a gyroscope, and an accelerometer. All of these hardware elements are valuable for detecting either head movement or observed blinking behavior. In particular, the sensors that we use for recording data pertaining to head movement include both the gyroscope and accelerometer, while the sensor that we use for recording observed blinking behavior is the IR sensor.

With respect to head movement, use of the gyroscope was considered important because it would help to differentiate between types of head movement observed from users. It would differentiate head

movements by providing knowledge regarding the direction of movement. The accelerometer was used in order to provide knowledge pertaining to the observed magnitude of a given head movement.

The IR sensor was utilized in order to provide information on blinking by performing multiple measurements of distance with the aid of infrared light (see Figure 4). These distance measurements were important because they could be used to determine whether or not the eye of a user was either opening or closing during a given period of time. Essentially the distance of a user's eye in relation to the IR sensor would decrease when their eyelids closed and increase when their eyelids opened.

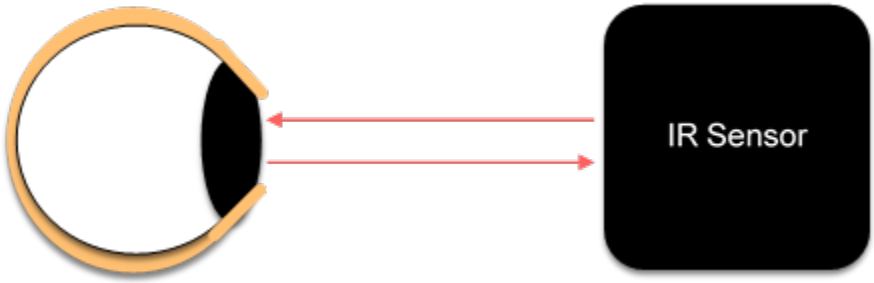

Figure 4: Use of the IR sensor for collecting blink data

There are a total of six different types of movement data, three from the accelerometer and three from the gyroscope. Additionally, the IR sensor was used to record two types of blink data. Those two types of blink data were SB and IB as defined in the background.

5.1.1 Data Collection Process

The data from the aforementioned sensors was collected from a user while a sequence of cue cards was shown to them on a visual display. This sequence, which repeatedly displayed cue cards in a given order was derived from content produced in a study [1]. In the study, the authors utilized a technique called rapid serial visual presentation (RSVP) when displaying content to users during testing. This technique was important for our display of visual content because it presented two different types of visual stimuli; a slide for directing user focus and a rapid display of millisecond-duration slides. The slide which directed user focus allowed us to collect SB data. The slides where a user searched for an image in the rapid display of millisecond-duration slides allowed us to collet SB + IB data (as the user is blinking normally and is also having blinks elicited from them during the sequence). Coupling the idea of RSVP with information known about how cognitive state influences blinking (see Table 1), it was considered possible to use RSVP for collecting two potentially distinct types of blinking behavior.

In addition to the use of focus slides and RSVP, instructional slides were also added before every SB + IB interval (see Figure 8) so that users would be in a more elevated cognitive state than they were during the preceding SB interval. Influencing the cognitive state of the user was regarded as an important task because cognitive state is a factor that has been established to influence blinking (see Table 1). Consequently, including cognitive state in the SB + IB stage was done in order to ensure that the observed blinking behavior collected for test subjects during SB and SB + IB intervals would be noticeably different. The exact implementation of the visual stage can be seen in Figures 5 and 6.

The preliminary instruction sequence shown in Figure 5 describes the sequence of the displayed cue card content that is run prior to performing a testing sequence with a user. After the preliminary instruction sequence is run, a data collection interval (see Figure 6) is displayed to the user while the system collects a single data point that is reduced to multiple features describing the user biometrics.

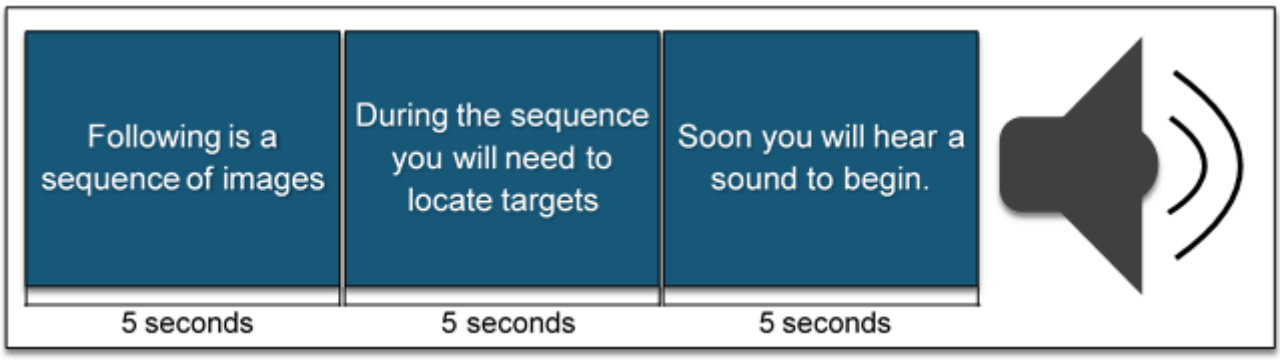

Figure 5: Instruction sequence during video collection

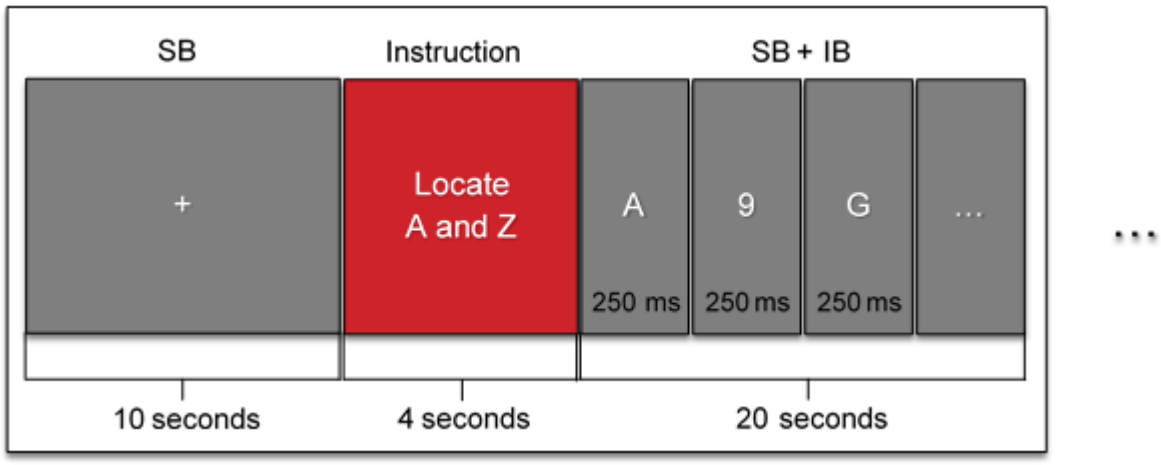

Figure 6: Interval timing sequence during video collection

After having collected the data from a user in the described way, biometric values of interest, otherwise known as **features** were then extracted from the raw data.

5.2 Data Processing via Feature Extraction

This section lists and describes the different features extracted from the two raw data types: blink data and movement data. We describe how each feature is unique and how we elicited the feature data from the raw data. There are 163 features collected from a set of user data (blink data and movement data). Appendix A contains a full list of these features. 162 features are elicited from the raw data and the 163rd feature is the user id, which identifies the user. See Figure 7 for where the data collection stage falls into the system design.

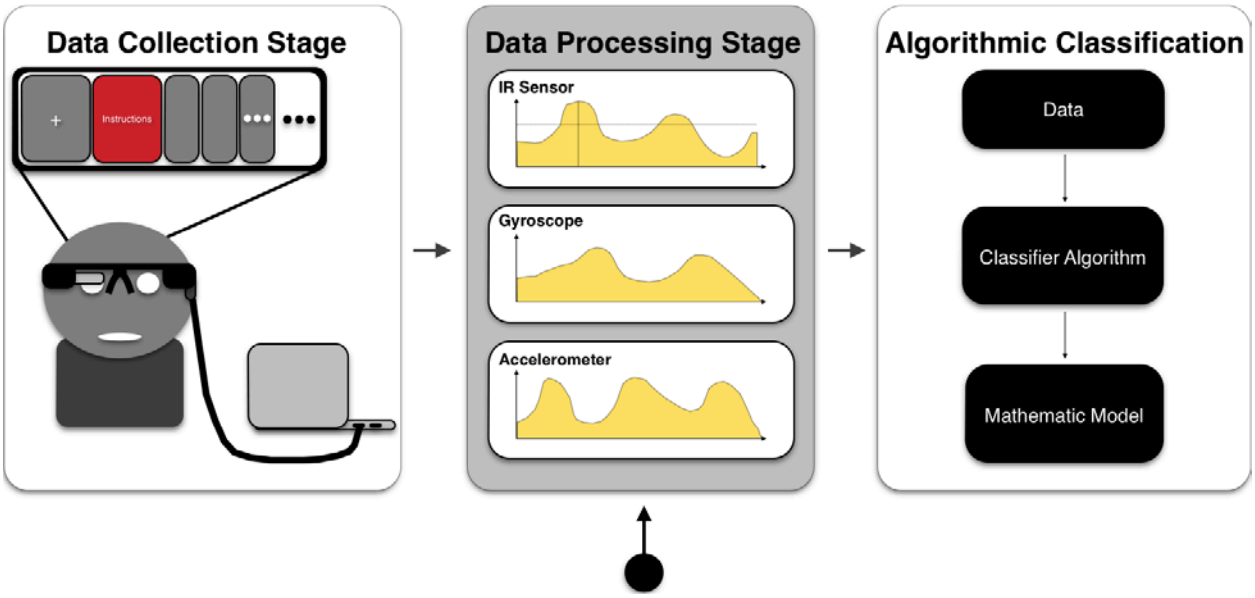

Figure 7: Data collection stage

For each type of measurement listed in this section, we record as features the average, minimum, maximum, standard deviation, variance and median of that value in the set. These are the six statistical measurements we choose to use to define some of the features.

We use a threshold to measure some of these features which is represented as a single standard deviation above the mean of the raw data values for the given dataset. To find where the start of a blink is, we set a threshold indicating the initialization of a blink. This will cut out a small part of the blink at the beginning and end of the blink, but as the same threshold will be applied to all collected blink data,

the actual time of the full blink is not as important to our research. The same threshold system is applied to head movements.

5.2.1 Blink Pattern Features

Blink data produces features related to lighting, blink timing and user physiology. For every blink feature listed below, there is a set of features related to SB data and a set related to IB data. We collect 36 features as described below from each set, giving 72 total features collected from blink data. A visual depiction of each feature is provided in Figure 8.

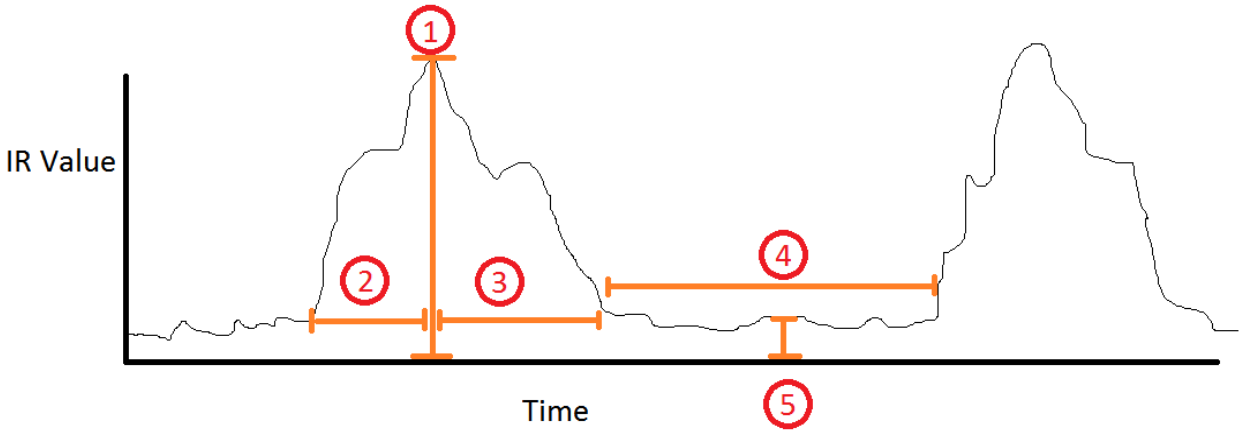

Figure 8: Features collected from blink data

Blink Pattern Feature: IR of Localized Blink Peaks (1)

This measurement is based on the recorded value of all aggregated, local peak measurements along the IR curve. Physically it translates to the IR value returned at the 'apex' of a blink, or when the user's eye is closed. This value should be unique because each user may have different IR values associated with their closed lid. As such, it should be a useful value to measure as a feature type as it should help narrow down the unique identity of the user. We collect 12 features from this collection using the six statistical measurements we previously mentioned over the IB and SB data types.

Blink Pattern Feature: Time to close eyelid on Blinks (2)

The time to close eyelid is the measurement from the start time of a blink to the fully closed eyelid; the halfway point of the blink. One user may typically have a faster down blink than another user and, in contrast they may open their eyes slower. As such, it could be a uniquely identifying biometric. We

collect 12 features from this collection using the six statistical measurements we previously mentioned over the IB and SB data types.

Blink Pattern Feature: Time to open eyelid on Blinks (3)

The third blink feature measures the second half of the blink from peak to eye-open time. This measurement is a pair with the time to close the eyelid feature. One user may open their eye slower or faster on average than another in differing environments, so this measurement could become a unique feature for identifying a user. We measure the feature from the localized peak of a blink until the IR value drops below a threshold indicating the blink is over. We collect 12 features from this collection using the six statistical measurements we previously mentioned over the IB and SB data types.

Blink Pattern Feature: Time between Blinks (4)

The time between blinks is the time measured between two blinks, from when one ends by the IR values dropping below the blink threshold, until the IR value passes above the threshold signaling the start of another blink. This feature could also be highly affected by environment. Users may have differing times between blinks as some people blink more often than others. We collect 12 features from this collection using the six statistical measurements we previously mentioned over the IB and SB data types.

Blink Pattern Feature: Minimum IR of Blink Pattern (5)

This is the measurement of the minimum IR of a blink pattern associated with the in-between blink phase. As there are long periods of minimum IR, we will be considering all IR values below the threshold representing the open eye state. This feature could be uniquely identifying as each user's open eye may produce significantly different IR values based on their environment. We collect 10 of the features from this collection using the six statistical measurements we previously mentioned over IB and SB data types. We exclude the maximum value in this measurement because there is a maximum threshold associated with this measurement.

Blink Pattern Feature: Total number of blinks

We measure the total number of blinks over the whole set as a single feature. This will indicate a user who blinks often or less often. This simple feature quickly demonstrates blinking pattern differences between users and is particularly useful for demonstrating differences between users' blink biometrics. We collect two features from this feature type, one for SB and one for IB.

Blink Pattern Feature: IR Value, undistinguished from blinks

For every IR value in the raw set of data we collect, we calculate the 6 statistical measurements as individual features (this would be the average IR value over the whole set, the standard deviation of IR values over the set etc.). These measurements are used over the whole, raw data set and are not affected by the presence of a blink. In this way, they may represent the trends in the data in a more abstract way that identifies a user better. We collect 12 features from this collection using the six statistical measurements we previously mentioned over the IB and SB data types.

5.2.2 Movement Pattern Features

Movement data is extracted into features describing **sudden head movements** (SHM) and **head movements** (HM). An SHM is when gyroscope movement data values exceed a threshold. An HM is when accelerometer movement data values exceed a threshold. We describe either an HM or an SHM as a movement.

Total, there are 90 features related to head movements collected by our system. 45 of these features come from SHM and 45 from HM. A visual depiction of each movement feature type is provided in Figure 9.

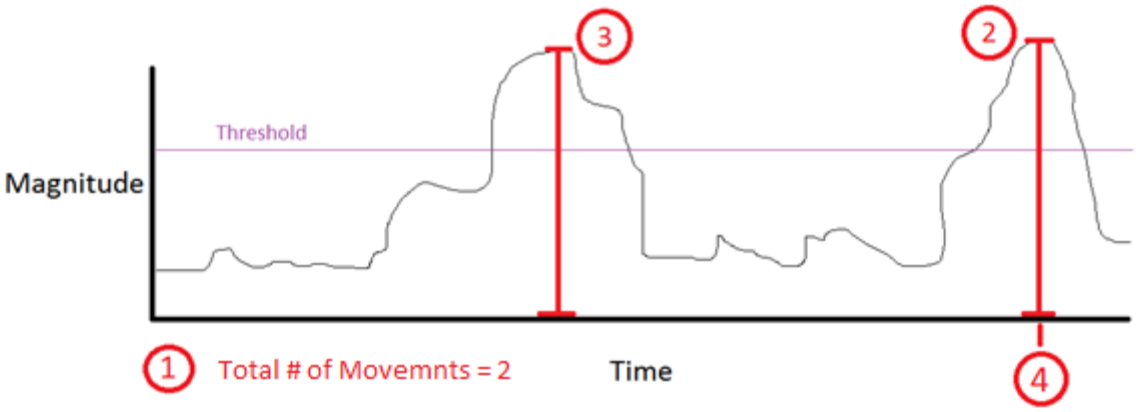

Figure 9: Features collected from movement data

Movement Feature: Total number of movements (1)

The total number of movements is measured over the collection period for one user. It can be uniquely identifying if a user has a particularly high number of movements during each collection, or if they have none. Movements may reflect the environment the user is in as well as user characteristics. Six features are collected from this feature type, one for each of the six movement data types.

Movement Feature: Peak magnitude of movements (2, 3)

This feature is the peak magnitude of all movements in a raw data set. It can be uniquely identifying in that some users may have extremely jerky movements while others may be more graceful or even still. We collect 36 features from this feature type using the six statistical measurements we previously mentioned over the six movement data types.

Movement Feature: Time of movement with greatest magnitude (4)

This is the time during the recording when the greatest magnitude movement was recorded. We may be able to identify if a movement regularly occurs during our collection for every user, or if a user has a unique habit of moving suddenly during the collection. We collect six features from this feature type, one for each of the six movement data types.

Movement Feature: Time of smallest movement

This feature mirrors the time of the movement with the greatest magnitude in that it records the time of the movement with the smallest magnitude. We collect six features from this feature type, one for each of the six movement data types.

Movement Feature: Magnitude, undistinguished

As with blinks, we also take the overall pattern of measurements into account by recording the six statistical measurements over the whole movement dataset. We collect 36 features from this feature type using the six statistical measurements over the six movement data types.

5.3 Machine Learning

To analyze the data we collect we will be using ML algorithms. ML algorithms can make a prediction of the outcomes of a circumstance based on having been trained on a large amount of data regarding similar circumstances. One could imagine the algorithms to be creating a high dimensional vector for each data point and relating them to each other so as to determine where the boundaries of classification are. In our case, every feature serves as a different dimension that the data point vector lies upon. Refer to Figure 10 to see where Machine Learning falls into our system design.

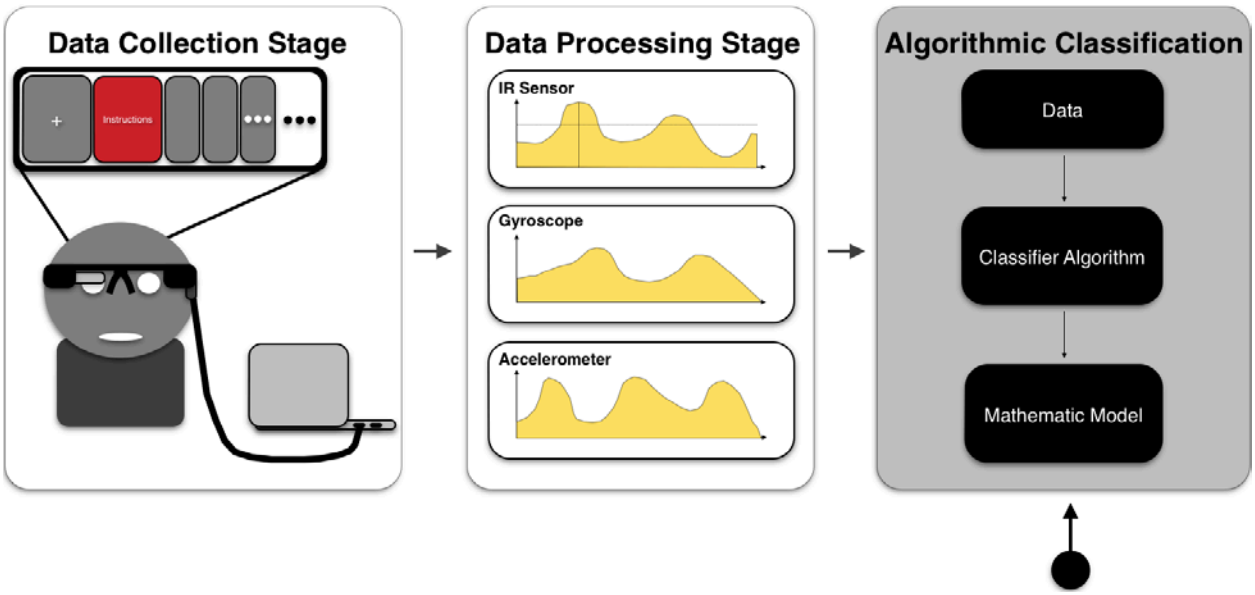

Figure 10: ML performance analysis

5.3.1 Feature Selection

The term **feature selection** in ML means tailoring a set of features to best fit an algorithm. One way this can be done is to run the desired algorithm with a **gain ratio evaluator** which evaluates a feature with respect to the ML algorithm. The **gain ratio** of a feature is the probabilistic gain the feature adds to determining the prediction of the outcome and is measured as a rank in comparison to other features. The higher the gain ratio, the more uniquely identifying the feature. The value of the rank that feature is given is between 0 and 1, 0 indicating the feature has no value and did not contribute to the prediction and 1 being a uniquely identifying feature that will always classify a data point correctly. Any feature with a rank above zero, we classified as **relevant** to the system analysis.

We decided that any feature rank above 0 would be deemed relevant because any positive rank could provide more accuracy to our system. Removing irrelevant features from algorithm analysis can improve performance by removing features that may be minutely throwing off calculations and putting more emphasis on features that have greater impact on a correct classification. Removing features also decreases the time needed to run a ML algorithm on a set of data.

We ran our data through the Functional Tree algorithm to get the gain ratio rank for each feature and determined which features were relevant. We used the Functional Tree algorithm because it had the best preliminary performance analysis. The top 5 features are referenced in Table 2. Notice that the most effective features were all blink related.

Table 2: Top 5 most relevant features

Rank	Feature	#	Type
0.698	147 SBLINK_MIN_IR_AVG	147	SB
0.694	119 SIBLINK_IR_AVG	119	IB
0.671	109 SIBLINK_MIN_IR_AVG	109	IB
0.664	102 SIBLINK_IR_MED	102	IB
0.662	151 SBLINK_MIN_IR_MAX	151	SB

We split this wholly relevant group of feature into different groups related to the feature's data type. The seven groups were: All features (108 relevant features), blink features (40 relevant features), IB features (23 relevant features), SB features (17 relevant features), movement features (68 relevant features), SHM features (33 relevant features) and HM features (35 relevant features). Each feature we collected fell into either the blink or movement category with respect to where the data came from; gyroscope, accelerometer, IR sensor. These seven groups, or features sets as we refer to them in our performance analysis, allowed us to understand what types of features had the greatest impact on the classification over different algorithms.

Table 3 depicts the total raw features in each features set and how many of those features were relevant (had a gain ratio above 0).

Table 3: Feature relevancy in comparison of feature sets excluding Username feature

	Total	*Relevant*	*Relevant %*
All	162	108	67%
Blink	72	40	56%
IB	36	23	64%
SB	36	17	47%
Movement	90	68	76%

Gyro(SHM)	45	33	73%
Acc (HM)	45	35	78%

The detailed list of relevant features will be provided in Appendix B.

5.3.2 ML Algorithms

There are many existing ML algorithms, each with different strengths and weaknesses depending on the data set they are run on. We describe the 3 main ML algorithms used to analyze our system performance to clarify their functionalities, strengths, and weaknesses in this section. These algorithms are Functional Tree, Multilayer Perceptron and Random Forest. A short description of each is provided below.

Functional Tree: Functional Trees fall under the ML algorithm category of classification trees. Classification trees are used to predict the value of a dependent variable (in our case, user id) by examining the effect (positive probabilistic rank) of each feature on the dependent variable and then sorting (predicting) the outcomes of unspecified data points based on the ranking of the features with regards to the value of that variable. The Functional Tree ML algorithm works well over many forms of data due to its flexibility in determining outcomes where as other algorithms have specific data requirements but may perform better [6].

Multilayer Perceptron: In ML, a multilayer perceptron classification algorithm trains a network of nodes that function in a manner similar to neurons in the human brain. Essentially, when a potential / threshold is reached for a given node (i.e. when sufficient conditions are met from a given weighting of connected nodes present in another layer) the node will fire (see Figure 11). Generally, the multilayer perceptron classifier algorithm is suitable for use in scenarios where it is advantageous to not make any assumptions relating to the use of probabilistic models. Consequently, they tend to perform poorly when a scenario can be more suitably modeled with the aid of probability. Due to the complexity of the Multilayer Perceptron algorithm, large feature sets take large amounts of processing power to train and predict information from.

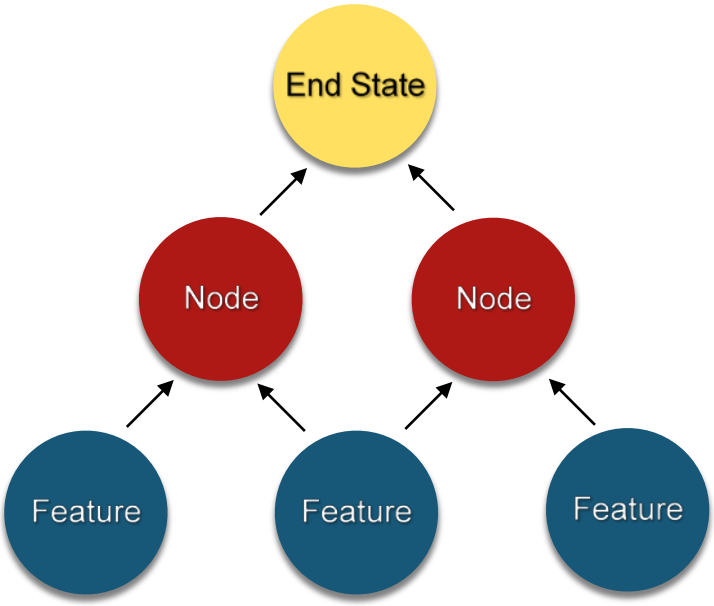

Figure 11: Multilayer perception algorithm

Random Forest: The Random Forest ML algorithm creates many simple tree predictors each constructed while considering a number of random features. The algorithm runs each test data point through each simple decision tree which classify the data point as a user in the user set. The forest (collection of trees) then vote on what each data point is classified as and the majority rules. Random Forest works better the more trees it has created. We ran the Random Forest algorithm with 100 trees and 7 features per simple decision tree. Random Forest may not work as well when there are more variations in classification to make (i.e. when there are more users to classify over) [13].

6 System Implementation

In designing our system, we began by selecting an HMD platform. We then designed a software-based system to capture relevant data and transmit it to a computer for storage and analysis. Figure 12 depicts our implementation of the system design stages.

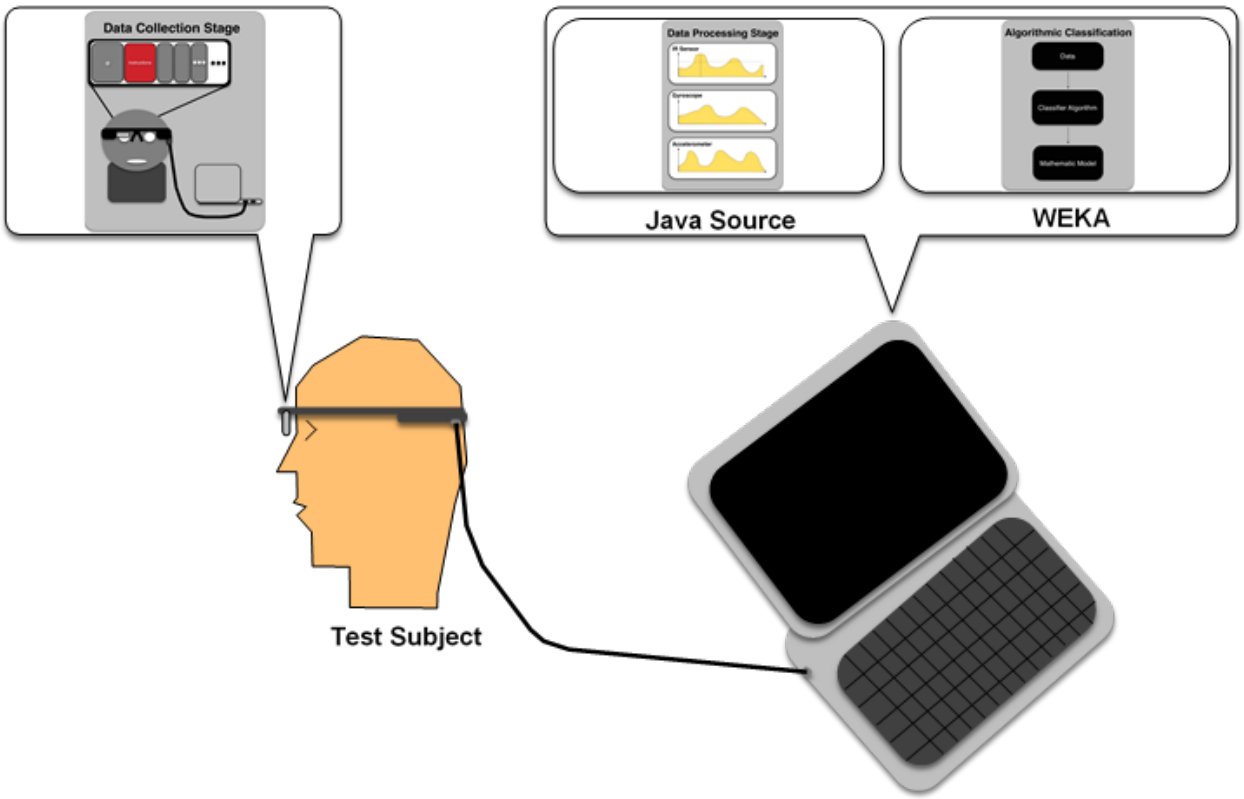

Figure 12: Implementation of design stages

6.1 Platform Evaluation

In order to select an HMD platform, the variety of sensors available on the HMD and their ability to collect biometric data was considered. Because the objective of our research was to apply both blink and head movement biometrics to user identification, only HMDs that were capable of detecting blink and head movement data were regarded as being viable candidates. Additionally, network capability was another consideration used when selecting an HMD. Network capability was considered as many HMDs have limited computational resources, which would mean transfer of data between the HMD and the computer would be necessary.

Because Google Glass was a commercially available HMD that possessed a collection of sensors capable of monitoring both head movement and blinking, it was selected as the platform of choice for our research.

6.1.1 Google Glass

The Google Glass comes equipped with all the standard sensors present in Android smartphones. It has a microphone, accelerometer, gyroscope, ambient light sensor, magnetometer, and camera. In addition

to this, it has an IR sensor for detecting when it is on a person's head which can also be used to detect blinks.

Built-in Sensor Operation

As the Google Glass runs a version of Android, we can interface with the sensors like in any other Android device. This involves using the Glass Development Kit (GDK) to setup a callback function to send us a notification whenever the system detects that the value of the sensor reading has changed for a specific sensor. When setting up this callback, a user may specify a relative frequency (Fastest, game, UI, and normal). This allows a user, ignorant of things like polling frequency, to effectively use the built in sensors.

Hardware

The heart of the Google Glass is a TI OMAP4430 processor [5]. This is a 32-bit processor developed by Texas Instruments [18]. The processor has access to 2GB of memory (1GB in earlier models) [17]. There are also 16GB of flash memory on the device, although only 12GB are usable by developers [17]. The device is powered by a 2.1Wh (approximately 570mAh) battery, which gives it a useful life of "approximately one day of regular use" [5][8]. The Google Glass is equipped with a camera that can capture either pictures at 5 Mega pixels or video at 720pixels [17].

A main sensor is an Invensense MPU9150 9-axis motion tracker [5]. It is composed of a 3-axis accelerometer, a 3-axis gyroscope, and a 3-axis magnetometer. The accelerometer outputs a 16-bit signal and can be configured to measure in the ranges +/-2g, +/-4g, +/-8g, and +/-16g. The gyroscope outputs a 16-bit signal and can be configured to measure in the ranges +/-250dps (degrees per second), +/-500dps, +/-1000dps, and +/-2000dps. The magnetometer outputs a 13-bit signal and only measures the range +/-1200micro-Tesla.

The Google Glass is also equipped with an IR sensor. This sensor can used to capture blinks [25]. The sensor does this by emitting infrared light [24][23]. The light sensor may also be used for detecting gaze direction.

Services on Google Glass

Google Glass uses the Android operating system. Android allows us to periodically query various sensors. For this project, we are mainly looking to use the accelerometer, gyroscope, and light sensor. In order to make good use of these sensors, we developed a set of tools that will effectively serve to

monitor the data that each of them collects. In particular, these tools take care of polling the sensors as well as storing and processing the data.

The Google Glass device is also able to transfer data between itself and a phone or computer. We preformed this task through a wired connection. There are two reasons for transferring data from the Glass to the computer. The first is that by offloading the sensor data to a phone or computer we could store it for later analysis. The other reason is that by using a phone or computer, we could perform more computationally intensive tasks to analyze the data for which the Glass is not well suited.

6.2 Software Architecture

Once the HMD platform was selected, a software-based system was then implemented on top of that platform so that data could be collected, transmitted, and analyzed. In order to develop this system, three distinct software components were required. The first software component, which was utilized during the data collection stage of our design, was the video software that supported the animation. The second software component was a mechanism that operated on the HMD platform to facilitate the transmission of data to a conventional laptop. The final software component was the server that operated on the conventional laptop in order to retrieve and store data collected from the HMD.

6.2.1 Data Collection Software Architecture

In order to implement use of the data collection stage on the platform of Google Glass, an Android application containing a single activity was developed.

Knowing that external events could occur and potentially disrupt the normal operation of an activity (in this case, the video animation and data collection) on the Glass, measures were taken in order to prevent the most common forms of undesirable Glass behavior. One of the measures taken while running the animation activity on Google Glass was to ensure that no other applications present on the device were used during testing. The other measure taken was to ensure that the power-saving / dimming functionality of Google Glass was intentionally disabled from within the activity whenever it was launched.

The video software architecture used for supporting the visual interface was first constructed by creating a standard Android Activity class in Java. After constructing the Android Activity class, the inherited `onWindowFocusChanged()` method was modified so that a viewport would appear visible during the runtime of the activity. In order to feed visual content into that viewport, two separate Android `animationDrawable` objects were produced by using two different XML files. Each line in

these XML files was used to specify both the resource ID of a particular image to be displayed as well as a duration over which to display the image. Once the `AnimationDrawable` objects were produced, the `animationDrawable` containing the preliminary instruction sequence was run while the second `animationDrawable` containing the sequence of data collection intervals was scheduled to run immediately afterward (see Figure 13).

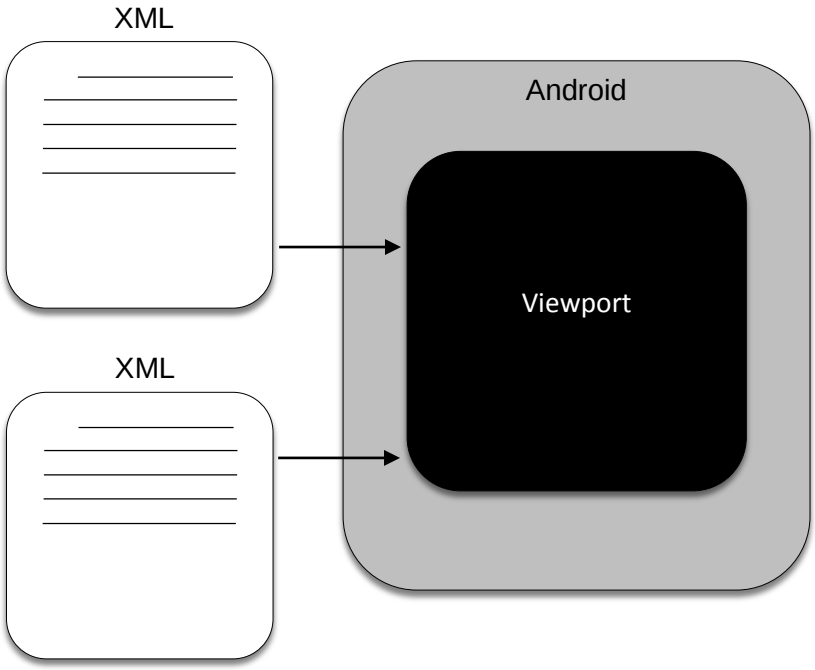

Figure 13: Using a viewpoint to show animation

Because the process implemented for transferring data was designed to coordinate with the video software, this communication was taken into account by providing a separate thread that would wait until the second `animationDrawable` would run. While the `animationDrawable` ran, the thread would then maintain a timed cycle that would be synchronized with the `animationDrawable` so that the sensor manager used for transferring data could be properly informed of when a new data collection interval was occurring.

In order to account for external factors that may prevent proper operation as mentioned previously, a flag against dimming / power-saving was set within the `onWindowFocusChanged()` method and the AndroidManifest.xml file was configured so that the heap allocated for images used while running each of the two `animationDrawables` was larger than the heap size that was allocated by default.

During the operation of the visual interface in the data collection stage, a multithreaded design was leveraged in order to both retrieve data from the sensors present on Google Glass and transfer that data back to a conventional laptop over a USB connection.

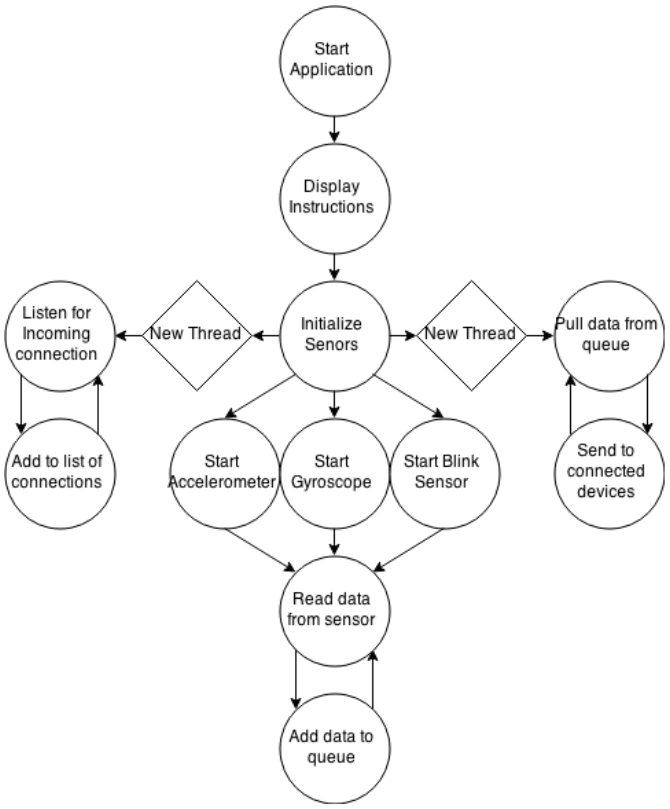

Figure 14: Process flowchart for Google Glass application

Figure 14 shows a rough schematic of the portion of the code that runs on the Google Glass. The actual program runs in many threads as indicated in the diamonds. Except for the blink sensor, the sensors use an interrupt mechanism instead of continuously polling. When launched, the program begins by showing a series of instructional images. After these images are shown, the application begins recording and relaying data. This moment is indicated by the start state.

The application begins by spawning a thread to handle incoming connections. For the transfer, the Google Glass acts as a client, accepting connections on a pre-established port, and sending the same information to all connected programs. This thread will continue to run and accept connections until the program ends or the user exits.

Next, the application initializes the sensors. The accelerometer and gyroscope are accessed using the built in interrupt system. Since this system uses the main thread, we minimize processing by simply appending data to a queue instead of immediately sending. The blink sensor operates by directly reading the sensor value from a file. This sensor runs in its own thread.

Finally, the application starts a final thread to handle sending the data to the computer. As long as it has at least one connection, this thread will send data from the queue to any connected computer. When the application terminates, it stops the sensors collecting data, finishes sending any data that remains in the queue, and closes any connections left open.

6.2.2 Data Processing Software Architecture

The computer side of the system is very simple. First, it sets up ADB to forward traffic on a specified port through the USB cable instead of sending it over a wireless connection. Then, it contacts to the application running on the glass over this port. When it receives notification that a collection has completed, it performs some basic processing of the data and writes it out to a file. Additionally, we adjust the timestamps of the data so that they begin at the start of a collection. Once the program receives the signal from the Glass that the Glass application has ended, or it goes several seconds without receiving any data, the server severs the connection.

Once the Glass has transferred the data to the computer, a `BlinkprintServer` on the laptop collects the raw data into a text file for later processing as seen in Figure 15.

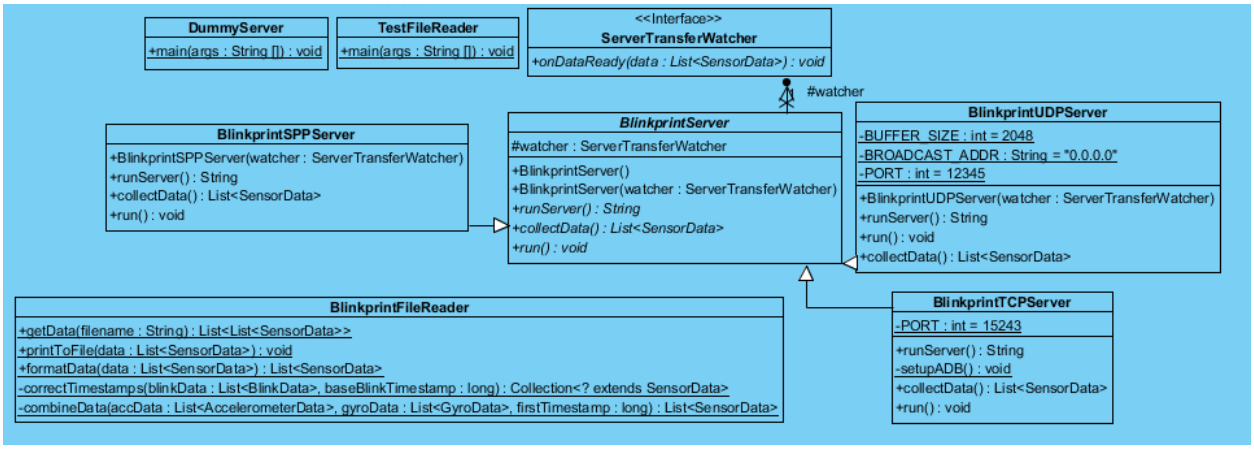

Figure 15: UML diagram for computer-side transfer code

Raw data from the Glass is sent to the server on the laptop which turns all of the raw data into feature data. The features are then compiled into an arff file, the type of file that is used by ML libraries to

predict the biometric capability of the data collected. A description of arff file output is given in Appendix E. The main arff creation class is called the `ArffBuilder` which generates the text and writes to the final arff file.

The `Arff builder` generates the `FeatureListFactory` as show in Figure 16.

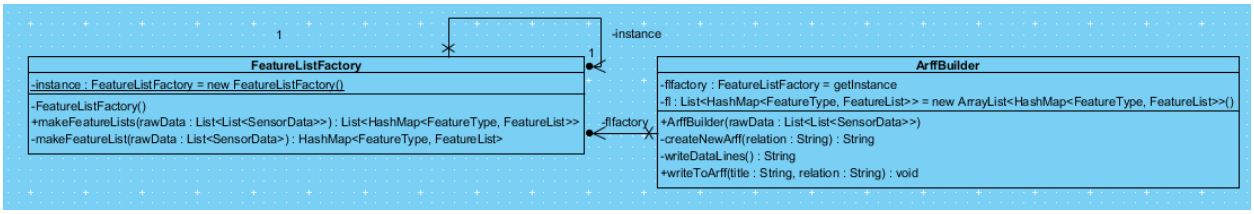

Figure 16: Creation of the FeatureListFactory from the ArffBuilder main class

The server receives a list of a list of sensor data. Each outer list is equivalent to one collection of user data. Each inner list is equivalent to one point of sensor data which can be blink data or movement data.

Blink data is sorted into a different list depending on the timestamp associated with it because we collect SB and IB type data at different times during a collection. The `ArffBuilder` uses a class called `MapBuilder` to do this. Each feature type is associated with the time that it starts via the `TimeFeaturePair` class which tells the `MapBuilder` what time stamp is associated with which feature types as seen in Figure 17.

Based on the Feature Type that the `MapBuilder` determines the raw data to be, it throws the data into an `IBlinkList`, `SBlinkList`, `MovementList` or `UserList`. In this way, once all of the data points have been processed, there are 4 advanced lists of data containing all of the raw data as seen in Figure 18.

Each concreate realization of a `FeatureList` has a parse function which outputs a map of `ArffFeatures` to `ArffAttributes` as seen in Figure 19. An `ArffFeature` is the enumeration of the feature that is collected from the `FeatureList`. An `ArffAttribute` contains the value of that feature.

The `ArffBuilder` then uses the `ArffFeature` to `ArffAttribute` mapping output by every `FeatureList` generated by the `MapBuilder` to generate the arff file. We then used an ML GUI to run our analysis on the data.

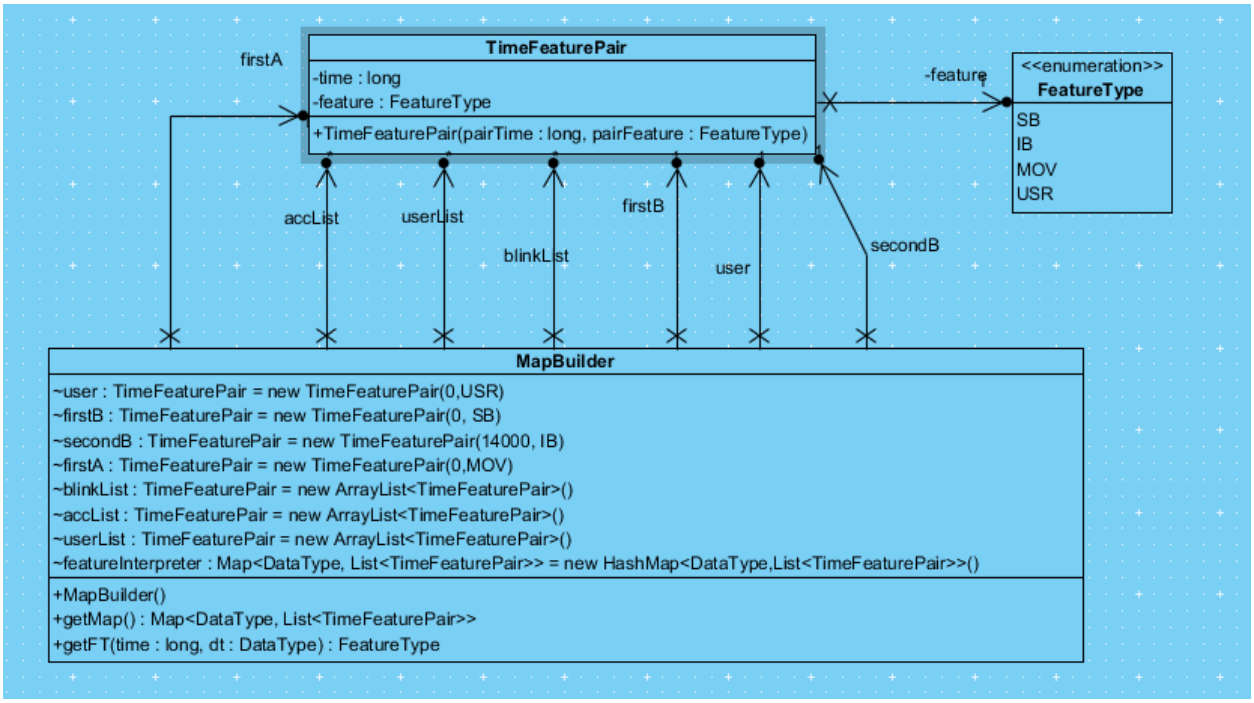

Figure 17: UML diagram of MapBiolder sorting raw data into FeatureLists

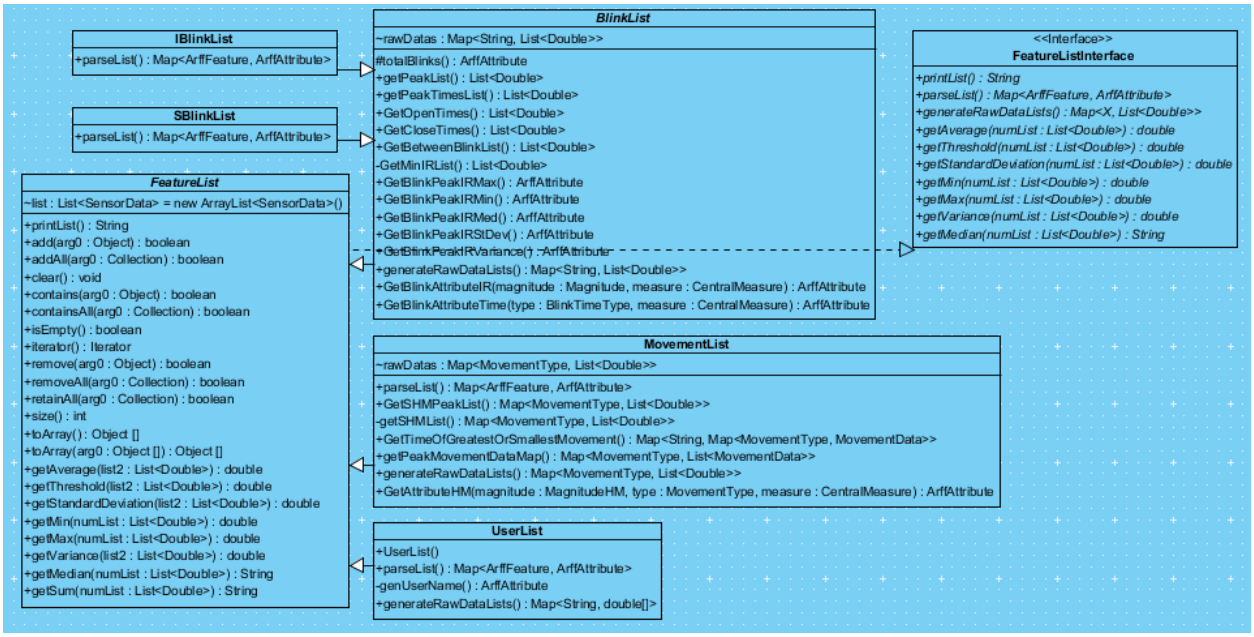

Figure 18: UML diagram of FeatureList concrete realizations

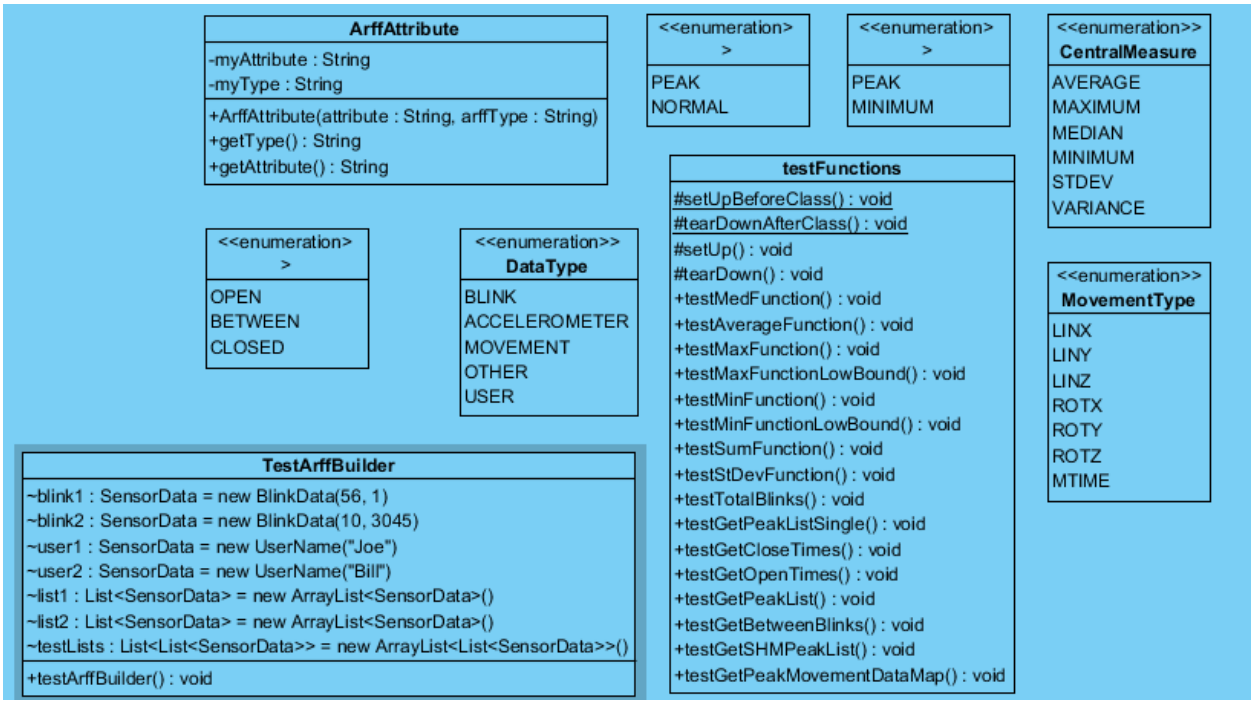

Figure 19: UML diagram of server enumerations

6.3 WEKA

The ML application of the **Waikato Environment for Knowledge Analysis** (**WEKA**) was selected for our research. We selected it because it provided a comprehensive suite of data preprocessing techniques which allowed us to flexibly manipulate our data. WEKA also provides a comprehensive suite of ML classifier algorithms which allowed us to explore which algorithms were best suited to our system. WEKA provides useful statistical output for the performance of trained classifiers so we could analyze system performance effectively. We used the WEKA GUI, WEKA Explorer, to preform our system analysis [22]

7 System Performance Analysis

To analyze the performance of our system, we collected data from 13 study participants. The ML algorithms used to analyze the system performance were Functional Tree, Multilayer Perceptron, and Random Forest. We determined that the Functional Tree algorithm was most appropriate for much of our analysis.

7.1 Performance Analysis Metrics

In our system, we used 10 fold cross validation to train and test our data. What this means is that the algorithm used ran 10 times, training itself on 90% of the data and testing its training on the other 10% of the data each time, eventually working through until it has predicted 100% of the data results. We used 10 fold cross validation so that we had the predicted outcome of each data point having the algorithm trained on 90% of the data and because it is a standard practice in ML. Evaluation of the ML algorithm is done by comparing the algorithm's prediction to reality.

There are many ways to analyze the performance of an algorithm. Much of the analysis comes from four values that help to sum up an algorithm's performance as seen in Table 4.

Table 4: Confusion matrix diagram

True Positive (TP)	False Positive (FP)
False Negative (FN)	True Negative (TN)

The **true positive** (TP**), false positive** (FP), **false negative** (FN) and **true negative** (TN) are major indicators of the performance of an algorithm. If we have a User A and a User A' which is any user who is not User A, the categories TP, FP, FN and TN respectively mean:

- TP: User A was correctly predicted to be User A
- FP: User A' was incorrectly predicted to be User A
- FN: User A was incorrectly predicted to User A'
- TN: User A' was correctly predicted to be User A'

Certain measurements can be gleaned from the confusion matrix values:

- Accuracy = TP + TN/ Total population

Accuracy defines the total number of correct predictions out of all predictions made. In our system accuracy indicates the number of times the system correctly predicted when User A was User A and when User A' was User A' out of the total number of predictions. Accuracy is good to show trends between ML algorithm performance but is not a great estimate of overall performance as it often

portrays a better performance than the system may actually have due to the massive amount of True negatives predicted correctly. Table 5 depicts the cumulative confusion matrix for our raw data to demonstrate the disproportionate TN predictions.

Table 5: Example confusion matrix

	Prediction	
Truth	TP: 346	FP: 44
	FN: 44	TN: 4636

- Precision = TP/ TP + FP

Precision indicates the ratio of the number of times the algorithm predicted User A was User A correctly over all of the times it classified any user as User A. Precision indicates how good the algorithm is at classifying a user positively (yes, it is User A). Low precision may indicate that the algorithm predicts too many things as positive, incorrectly (trigger happy algorithm).

- Recall = TP / TP+FN

Recall is considered a companion performance measurement to precision. In our system, recall represents how often a User is predicted to be User A when they actually are User A. Recall considers how good an algorithm is at predicting the truth.

- F-Measure (Alternatively, F-score)= 2TP/(2TP+FP+FN)

The F-measure is the harmonic mean of the recall and precision. It is a good indicator of overall algorithmic performance because it omits TN.

- FP Rate = FP/ TN+FP

FP rate is the inverse measurement to recall. FP rate represents the percentage of times the algorithm classified a User as User A when they indeed were User A'. As FP rate is an inverse of recall, a lower FP rate indicates better algorithmic performance. Intuitively, in our system, a high FP rate would indicate a system which incorrectly identifies users often.

- FN Rate = FN/FN+TP

FN rate is the inverse measurement to precision. FN rate represents the percentage of times User A was classified as User A'. Again, As FN rate is an inverse of precision, a lower FN rate indicates better algorithmic performance. A high false negative rate in our system would mean the system rejecting users because they were identified as not themselves often.

We used these ML performance metrics on varying user data sets.

7.2 User Data Collection Process

In addition to our three team members, we collected data from ten WPI students. Before beginning the experiment, we ensured that the Google Glass and testing computer were charged, and that we had access to a quiet space with adequate fluorescent lighting. We also prepared the Google Glass to collect and transmit data. The WPI students were informed of the purpose of our study and asked to sign a consent form approved by the WPI IRB. The consent form can be found in Appendix C. The background data on relevant anonymous test subjects can be found in Appendix D.

We recorded the student's gender, age, and whether or not they wear contacts or glasses. As glasses interfere with detecting blinks, we asked any subject with glasses to remove them and either wear contacts or verify that they could see the Google Glass screen clearly enough. We then put the Google Glass on the subject and adjusted the device so the subject was comfortable and could see the whole screen. Finally, we ran the Glass application and collected the required number of data-points. A detailed step-by-step methodology can be found in Appendix F.

Notably, we split the data up into different test groups: group with 3 to 13 users, 30 data points each, and a group of three users with 90 data points each. Comparing the groups of users that each have 30 data points per user allows us to see the effect of the amount of users in a test group on system performance. Comparing the group of three users with 30 data points to three users with 90 data points will allow us to determine the effect of an increased amount of data points on the system performance.

7.3 Determining Effective Algorithms

We determined that the Functional Tree algorithm was the most efficient algorithm to use in our system analysis due to its high F-measure on relevant data and speed of processing.

To determine which three algorithms we wanted to test, we needed to find the top performing algorithms on our relevant data set. We ran all of the applicable algorithms available in WEKA on the test with 13 users, 30 data points each, without removing irrelevant features. We compared the F-

measure of all of these ML algorithm's results to determine which algorithms had performed the best, the higher being better. The three ML algorithms with the highest F-measures were Functional Tree, Multilayer Perceptron, and Random Forest. A graph of the top five algorithms and their respective F-measure scores can be found in Figure 20.

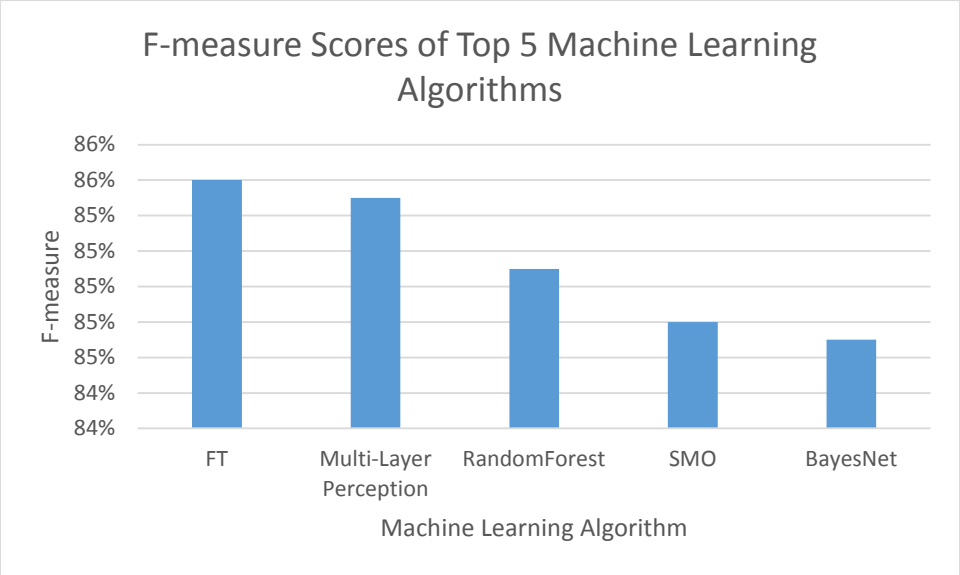

Figure 20: F-measure scores of top 5 ML algorithms

Once we had determined the top three performing algorithms on the data set which included irrelevant features, to determine which of the three was best suited to the system, we reviewed the other performance statistics starting with accuracy. Note that while Functional Tree did get a higher F-measure in Figure 20, in Figure 21, it had a lower accuracy.

We then compared all of the performance scores listed in section 7.1 to determine which algorithm most suited our system for further analysis.

Figure 22 demonstrates that Random Forest performs worse compared to Functional Tree and Multilayer Perceptron in each performance measurement. All performance metrics indicate a significant trend toward being able to identify a user.

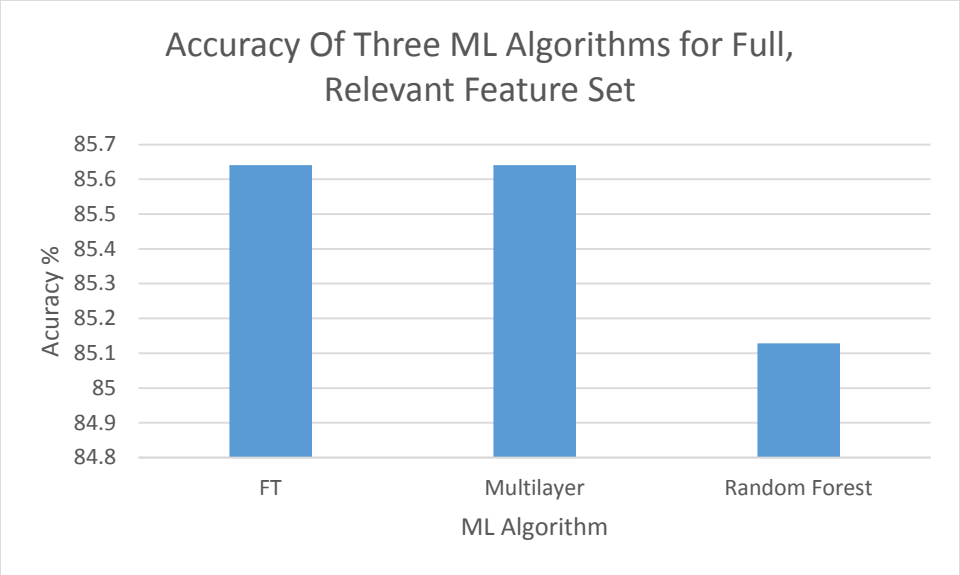

Figure 21: Accuracy of three ML algorithms for the full, relevant feature set

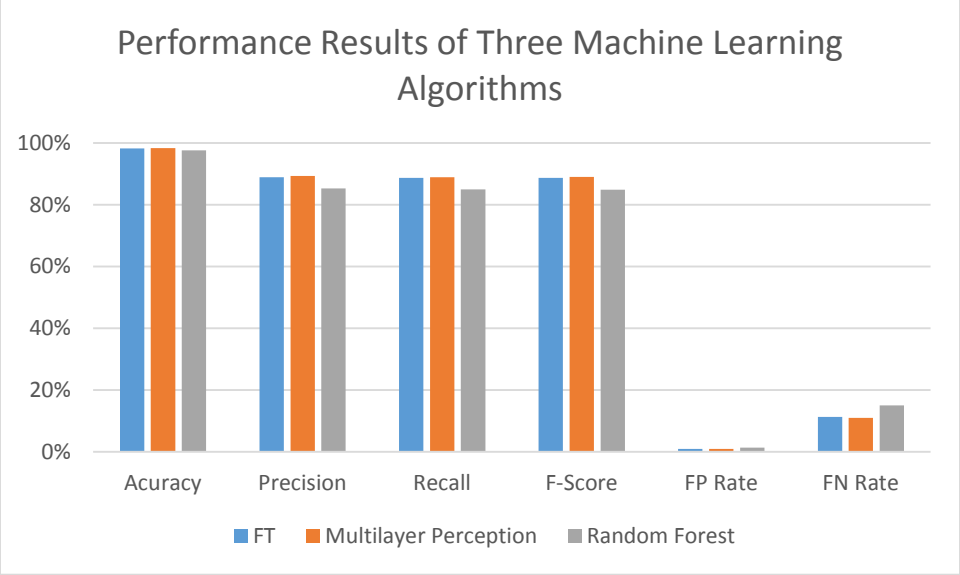

Figure 22: Comparison between performance results for three ML algorithms

Table 6: Performance metrics of Functional Tree, Multilayer Perceptron and Random Forest

	Accuracy	*Precision*	*Recall*	*F-Measure*	*FP Rate*	*FN Rate*
FT	98%	89%	89%	89%	1%	11%
Multilayer Perceptron	98%	89%	89%	89%	1%	11%
Random Forest	98%	85%	85%	85%	1%	15%

Table 6 demonstrates that Multilayer Perceptron and Functional Tree had identical results. This tells us that using Multilayer Perception or Functional Tree would be an acceptable approach. How acceptable, is the question.

Precision indicates how often User A will be predicted as user A. Functional Tree and Multilayer Perceptron both will identify User A as User A 89% of User A predictions. 89% is a significant figure considering there are 13 users in the database and indicates that both algorithms can identify users successfully.

A False negative result of 11% would mean that the identification system would positively identify 11% of false users. As our system aims to identify users to better system performance, 11% is an acceptable measure because a false identification is an annoyance to the user, but not particularly harmful as they could just attempt to re-identify.

As both Functional Trees and Multilayer Perceptron are comparable in terms of statistical results, it should be noted that Multilayer Perceptron takes 10 times longer to run than the Functional Trees algorithm on our data. The Functional Tree F-measure is slightly higher than the Multilayer Perceptron F-measure as well. As such, we considered Functional Tree as the optimal algorithm for identifying users over the full relevant feature set. Our next goal was to understand the effect of differing features on results.

7.4 Determining Effect of Features Selection

We separated our data into seven different feature sets as explained in section 5.3.1. We then ran the Functional Tree classifier on the differing feature sets to determine the effect of differing feature types on system performance.

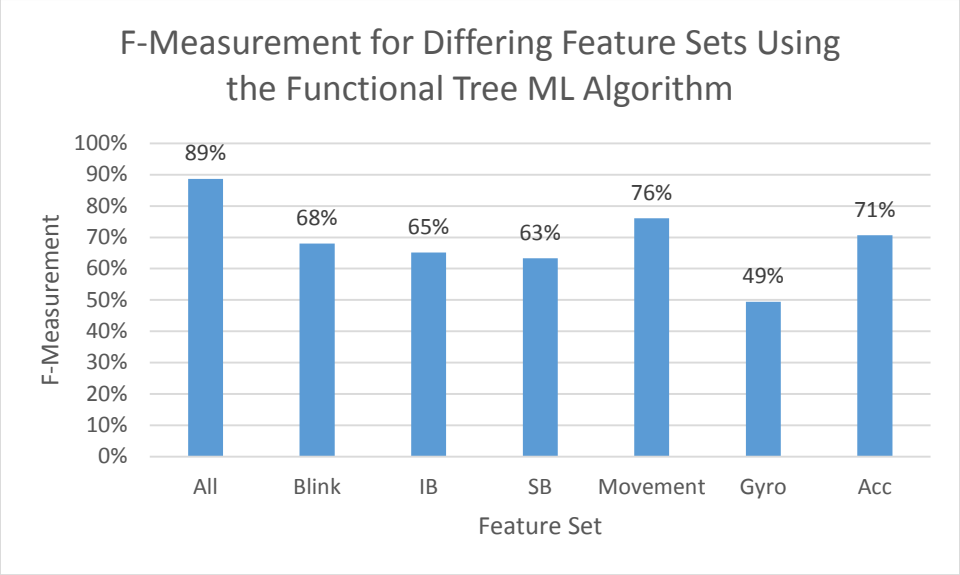

Figure 23: F-Measurement for different feature sets using the Functional Tree ML algorithm

Figure 23 demonstrates that the movement feature set had the greatest F-measure of all of the feature sets. Out of all of the data types (SB, IB, Gyro (SHM), Acc (HM)), acceleration had the highest F-measure which indicates that it may serve as a more efficient biometric identifier than the other feature types. In an application of this system with limited processing power, modeling the most efficient features will be necessary.

The Gyro feature set had the lowest F-measure of 49%. This means that nearly 50% of the time, the gyro feature set incorrectly predicted the correct user associated with a specific data point. Interestingly, adding the Gyro feature set to the Acc feature set did increase the performance of the algorithm by 5%. A system with high processing power can include feature types like Gyro because they do increase the overall performance of the algorithm at a high cost which is negligible when performance is not considered expensive.

Both blink sets are reasonably comparable, showing slightly higher F-measure IB in comparison to SB. The blink feature sets scoring above 50% does indicate that one can identify a user using just their blinks to an extent.

Other performance metrics of differing feature sets over Functional Tree are shown in Figure 24. Numerical data for this chart is shown in Table 7. Note that the Gyro feature set had a very high FN Rate meaning that in reference to the system it would identify a user incorrectly over 50% of the login attempts. Overall, FN rate and FP rate increases dramatically when the amount of features used is decreased. All of the feature groups have an unacceptably high FN & FP rate which indicates that only with all of the feature sets combined does the system operate effectively.

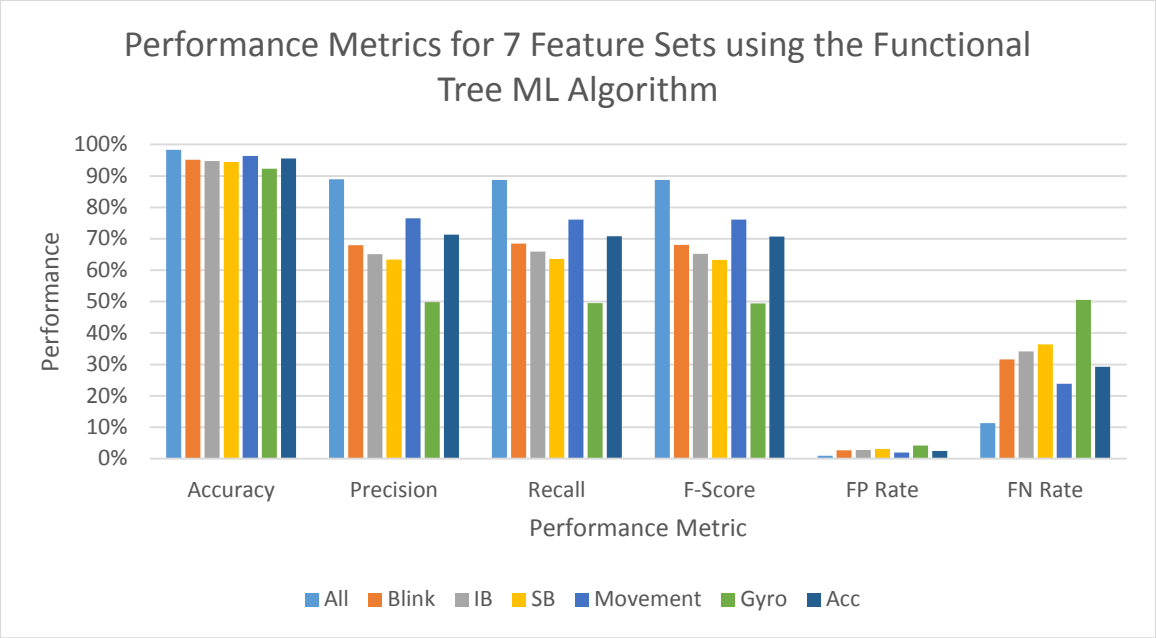

Figure 24: Performance metrics for 7 feature sets using the Functional Tree ML algorithm

Returning to the analysis of Multilayer Perceptron and Random Forest with respect to differing feature sets seen in Figure 24, one can see that while the Functional Tree algorithm does best in the case of all attributes, the Random Forest does better in cases where Multilayer Perceptron and Functional Tree do not perform as well, specifically in all feature sets that do not include all of the feature types. In a system with low processing power, it would be ideal to use the Random Forest algorithm over Functional Tree or Multilayer Perceptron. There is no case where Multilayer Perceptron preforms better than Functional tree in Figure 25.

Table 7: Table of performance measurements for 7 feature sets using the Functional Tree ML algorithm

Feature Set	Accuracy	Precision	Recall	F-Measure	FP Rate	FN Rate
All	98%	89%	89%	89%	1%	11%
Blink	95%	68%	68%	68%	3%	32%
IB	95%	65%	66%	65%	3%	34%
SB	94%	63%	64%	63%	3%	36%
Movement	96%	76%	76%	76%	2%	24%
Gyro	92%	50%	49%	49%	4%	51%
Acc	96%	71%	71%	71%	2%	29%

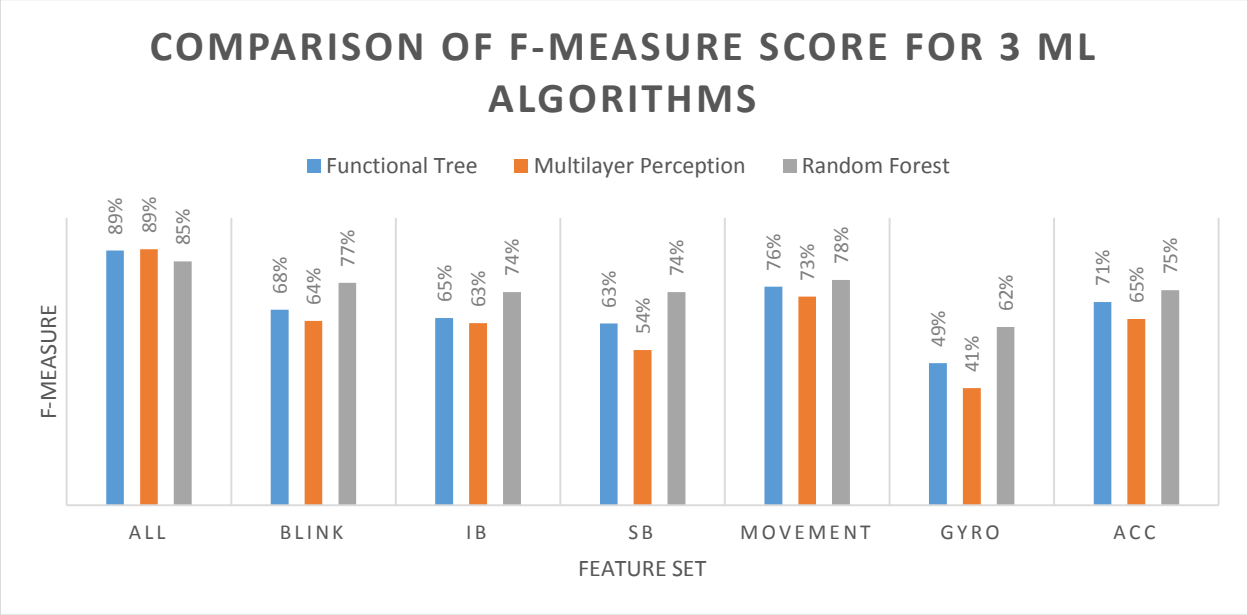

Figure 25: Comparison of F-Measure score for 3 ML algorithms over different feature sets

Our analysis demonstrates that the Movement feature set has the highest potential for biometric identification, but only the whole feature set has an acceptable error performance. The Random Forest

algorithm preforms better overall for smaller feature sets. The next goal was to determine the effect of the size of the user set (how many different users participated in the study) on performance.

7.5 Determining Effect of the Amount of Users in Data Set

To determine the effect of the amount of users in a dataset on system performance, we ran the ML algorithm Functional Tree over data sets of 3, 5, 7, 9, 11 and 13 users. Each of these data sets is a sub-set of the original 13 users (we did not collect new data for this experiment). We compare the performance results of the datasets to determine the effect of the amount of users on the performance of the algorithm.

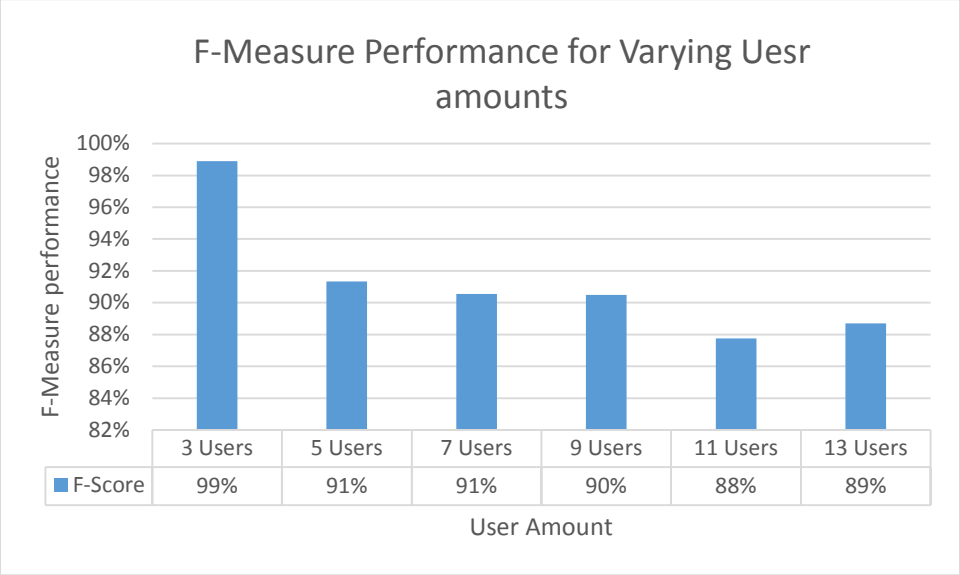

Figure 26: F-measure performance for data sets with 3, 5, 7, 9, 11, and 13 users

Figure 26 demonstrates the results of running the Functional Tree algorithm on these data sets which each have 30 data points collected per user. Unsurprisingly, performance generally improves when there are fewer users in the user set. Notice how performance drastically increases from an F-Measure of 91% to an F-Measure of 99% between 5 and 3 users. As well, the F-Measure drops 2% from 9 to 11 users but increases 1% from 11 to 13 users. This may be due to the specific variation in the different user groups (i.e. some user groups will be more similar while others will have very unique users allowing for better F-Measures. In terms of implementation, on a system with very few users who log on often (say, a family-shared device), our system works well in terms of identification. Figure 27 shows detailed performance metrics for the user variance study. These performance metrics indicate similar trends where data sets with less users perform better overall.

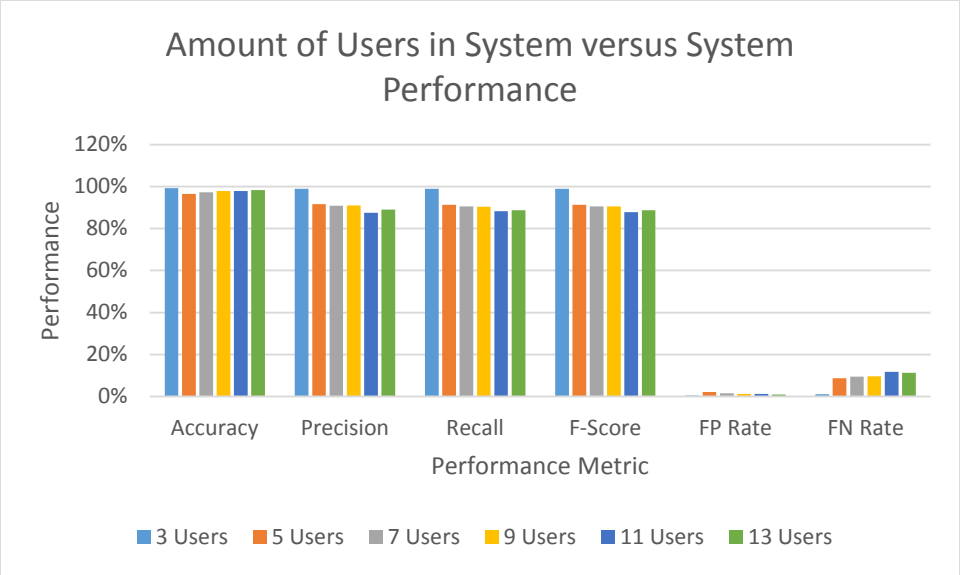

Figure 27: Performance metrics for data sets with 3, 5, 7, 9, 11, and 13 users

The amount of users in the data base will vary the performance of the system. Another consideration with ML is how much data will each user need to be predicted well. Determining the effect of the amount of data supplied per user to identify them is crucial to analyzing system performance.

7.6 Determining Effect of Data Amount on System Performance

We had collected over 90 data points from three of the student test subjects (the researchers) which allowed us to compare the results of Functional Tree run on the three test subject's data with 30 data points and with 90 data points.

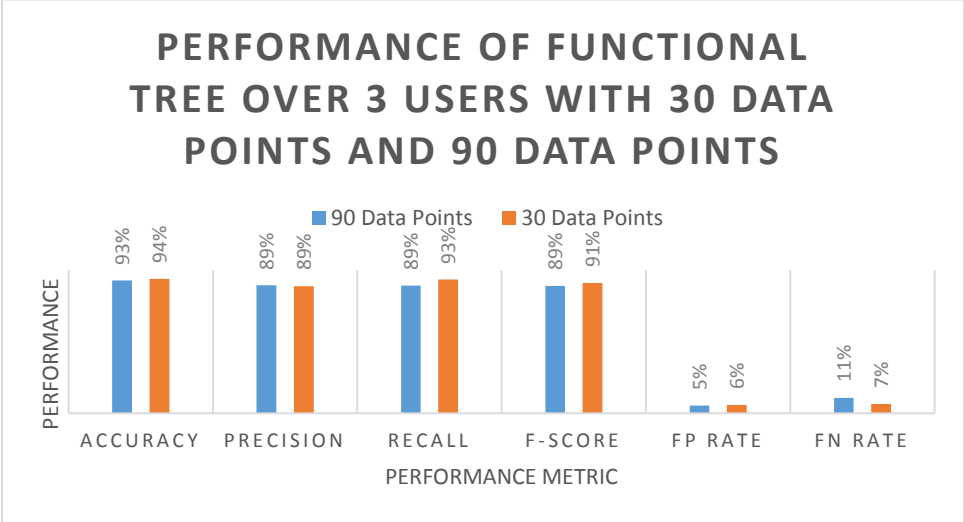

Figure 28: Performance of Functional Tree over 3 users with 30 data points and 3 users with 90 data points

Figure 28 depicts the performance results between the two sets. It would intuitively make sense to say that with more information, an algorithm would perform better. The data does not support this hypothesis. The F-Measure for the 30 data point set is 2% higher than for the 90 data point set. The only metric that the 90 data point set does better is FP Rate by a single percentage point. In the case of three users in the data base, it seems sufficient to identify them with less data points. In fact, the system seems to work better when there are less data points per user in the case of having three users in the system. We tested varying data points over only three users which is a particularly small user set. Results may change with more users.

8 Discussion

Our results could have been compromised by some of our testing procedures. We also could be omitting major features that affect a user's blink and head movement pattern.

Test setup: The flow of data collection was not consistent for experiments. The glass over-heated at times causing all the results of that data collection to be null and compromised. The connection between the computer and the glass would sometimes break mid-collection, compromising the last data point collected.

Omitted Features: We omitted some features that one could theoretically collect from an HMD which may have had influence on the user's blinks. The time of recording could have relation to the drowsiness or alertness of the user. The times we collected the data at were sporadic and may have had an influence on our data. The location and weather of said location could also be collected. The date may also be relevant to the season; a temporal study would be needed to determine this.

Performance Analysis: Our analysis of the effect of the amount of data points per user was inconclusive due to the lack of data we had to define larger user groups. We can't conclude that the amount of users in this data set did not have an effect on the results here.

9 Conclusions

Our results demonstrated that identifying a user with blink and movement features collected from a HMD is possible. Our approach demonstrates that using many different ML algorithms, a user could be identified in a system based on features related to their blink and their head movements. Head movements were more biometrically identifying than blink features.

While Google Glass has been currently triaged by Google, there is a growing market for HMDs that may function in a similar enough manner to Glass that the application of our research could become the basis for head-mounted user identification.

10 Future Work

While the research presented in this paper has successfully demonstrated that user identification with HMDs is feasible, there are a variety of additional measures that can be taken in order to improve the performance, mobility, utility, and practicality of the developed system.

One of the first experiments that should be run is to determine exactly how the system responds to different numbers of users and different numbers of data points for each user. This would include tests using very large numbers of users with only a few samples, and a very small number of users with very large number of samples. This type of information is useful in determining what type of applications the system can be used for. It would also allow a future team to tune the collection process. If it turns out that it is possible to ensure accuracy with only ten data points, then the collection period could be cut down drastically. This experiment would also demonstrate the boundaries of the model constructed in this paper. This would be important for both decreasing the time required for identification and understanding the rate at which accuracy in identification degrades.

Before the system is turned into an actual identification system, work needs to be done to analyze the long term stability of a person's natural pattern of blinking. Our data was collected in a reasonably isolated location during a period of about twenty minutes. This left little opportunity for a person's pattern of blinking to change. To test if a user's blink changes over a period of time, we could take data from a single subject over a period of several months. Data could be taken at different times of day as well to look for any natural patterns of change during the day. This could show that a person's natural pattern of blinking stays within a reasonable tolerance over a period of time.

Another measure that can be taken to improve performance would be to modify the design of certain software components in our system so that the process of data collection is more streamlined. Currently, the system requires a large amount of resources due to the ML algorithms. The current system uses too much processing power to stand alone on most current HMDs. The amount of resources consumed could potentially be reduced by improving the efficiency of the multithreaded application, and reducing the quantity or quality of images allocated in memory for the visual interface. Additionally, the system currently requires the user to start two programs, one on the computer and

one on the Glass. While this is not very difficult, it can be annoying and would not be tolerated in a real world application. Instead, a more user friendly solution must be found. One possible solution would be to have the authentication system on a phone or computer constantly running as a background service. Instead of being started by the user, it would listen for any connection from the Google Glass and only activate then. Alternatively, the Google Glass application could be set up to exist in the background and only come to the front when a user requests it to from their phone or computer.

In order to make the system into something useable in the real world, we would need to transition away from having the Glass connected to the computer by USB cable. The most useful way to do this would involve porting the processing performed on the computer to a smartphone. Instead of using a USB cable, we would use the existing Android libraries for communicating over Bluetooth. This connection can be encrypted to protect the data from an attacker. One of the biggest challenges in doing this would be getting the ML algorithms to run efficiently on a device with limited processing power. Likely, we would need to work to reduce the amount of data needed so as to reduce the total amount of processing the algorithm needs to do. In order to improve the utility of our system, there is a need to understand how the model developed by our system can be integrated with the security interface implemented in the Android operating system.

The practicality of our system could also be improved by accounting for non-deterministic events that frequently occur on mobile platforms. These events often include unexpected program termination due to resource consumption and transitions in foreground that are directly caused by activity-related events such as phone calls. If a person frequently gets messages or calls, it may be difficult to get enough complete data to successfully authenticate them. It may be possible for a system to handle cases such as this by salvaging the incomplete data. Maximum and minimum values are unlikely to change too much if a user has already completed half of the authentication period. It should also be possible to extrapolate from the existing data to get an estimate for the number of event type features. Successfully handling such events during runtime is a software engineering task that must be performed in order to fully realize the deployment and adoption of our system in a real-world environment.

Additionally, since the primary concern of our system is security, threat modeling must be performed in order for our system to be made practical enough for deployment. Some potential considerations in this area pertain to establishing a cryptographically secure channel of communication between a mobile device that is running our server-side code and an HMD that is running our system. Another security consideration would be to prevent other mobile devices from interfering with the operation of our

system, and to ensure that other applications cannot be leveraged to corrupt either our software or the database that it relies upon for identification.

Finally, it would be useful to be able to control some of the inherent randomness in a person's blink patterns. It may be interesting to add additional variables related, not to the user, but to the environment. This would include information such as light level, noise level, time of day, pollen count, etc. Light level can effect measurements we take such as IR blink peak. Keeping track of ambient light could help improve accuracy even when the user is using the system in multiple locations, or moving while using the system. Cognitive state affects the frequency of a person's blinks. Knowing the time of day could indicate whether a user is tired or just woke up. Finally, a high pollen count can increase the blink frequency of users who suffer from seasonal allergies.

11 References

[1] Abelson, M. B., & ORA staff. (2011). It's Time to Think About the Blink: In addition to defending the eye from particles and allergens, the blink can help diagnose certain diseases as well. Review of Ophthalmology.

[2] Abrahams, V. C. (1977). The physiology of neck muscles; their role in head movement and maintenance of posture. Canadian journal of physiology and pharmacology, 55(3), 332-338.

[3] Bentivoglio, A. R., Bressman, S. B., Cassetta, E., Carretta, D., Tonali, P., & Albanese, A. (1997). Analysis of blink rate patterns in normal subjects. Movement Disorders, 12(6), 1028-1034.

[4] Colzato, Lorenza S., Heleen A. Slagter, Michiel MA Spapé, and Bernhard Hommel. "Blinks of the eye predict blinks of the mind." Neuropsychologia e6, no. 13 (2008): 3179-3183.

[5] Google Glass Teardown. (2014) Catwig. Retrieved from http://www.catwig.com/google-glass-teardown/

[6] How To Predict Membership, Classification Trees. (n.d.). Retrieved March 9, 2015, from http://www.statsoft.com/Textbook/Classification-Trees

[7] Hogben, G. (2010). ENISA Briefing: Behavioral Biometrics. ENISA. Retrieved August 7, 2014, from https://www.enisa.europa.eu/activities/risk-management/files/deliverables/behavioural-biometrics

[8] InvenSense. (2013) *MPU-9150 Product Specification Revision 4.3*

[9] Ishimaru, S., Kunze, K., Kise, K., Weppner, J., Dengel, A., Lukowicz, P., & Bulling, A. (2014, March). In the blink of an eye: combining head motion and eye blink frequency for activity recognition with google glass. In Proceedings of the 5th Augmented Human International Conference (p. 15). ACM.

[10] Nakamori, K., Odawara, M., Nakajima, T., Mizutani, T., & Tsubota, K. (1997). Blinking is controlled primarily by ocular surface conditions. American journal of ophthalmology, 124(1), 24-30.

[11] Ousler 3rd, G. W., Abelson, M. B., Johnston, P. R., Rodriguez, J., Lane, K., & Smith, L. M. (2014). Blink patterns and lid-contact times in dry-eye and normal subjects. Clinical ophthalmology (Auckland, NZ), 8, 869.

[12] Patel, S., Henderson, R., Bradley, L., Galloway, B., & Hunter, L. (1991). Effect of visual display unit use on blink rate and tear stability. Optometry & Vision Science, 68(11), 888-892.

[13] Random Forests. (n.d.). Retrieved March 9, 2015, from http://www.statsoft.com/Textbook/Random-Forest

[14] Prasuethsut, L. (2015, March 6). Project Morpheus review. Retrieved March 8, 2015, from http://www.techradar.com/us/reviews/gaming/project-morpheus-1235379/review

[15] Shultz, S., Klin, A., & Jones, W. (2011). Inhibition of eye blinking reveals subjective perceptions of stimulus salience. Proceedings of the National Academy of Sciences, 108(52), 21270-21275.

[16] Simonite, T. (2015, January 21). Microsoft's New Idea: A Hologram Headset to Rewrite Reality. Retrieved March 8, 2015, from http://www.technologyreview.com/news/534356/microsofts-new-idea-a-hologram-to-rewrite-reality/

[17] Tech Specs. (2014) Google. Retrieved from https://support.google.com/glass/answer/3064128?hl=en

[18] Texas Instruments. (2014) SWPU231AP Technical Reference Manual

[19] The All New Oculus Rift Development Kit 2 (DK2) Virtual Reality Headset. (n.d.). Retrieved March 8. 2015, from https://www.oculus.com/dk2/

[20] Tinker, M. A. (1949). Involuntary blink rate and illumination intensity in visual work. Journal of experimental psychology, 39(4), 558.

[23] Weka 3: Data Mining Software in Java. (n.d.). Retrieved March 9, 2015, from http://www.cs.waikato.ac.nz/ml/weka/

[23] Westeyn, T., Pesti, P., Park, K. H., & Starner, T. (2005). Biometric identification using song-based blink patterns. Proc. HCI Int'l'05.

[24] Wheeler, A. J., Gomez, L. R. P., Raffle, H. S. (2013), US8611015 B2. Google Inc.

[25] Wink – Google Glass Help. (2014) Google. Retrieved from https://support.google.com/glass/answer/4347178?hl=en

[26] Witten, I., & Frank, E. (2011). Data mining: Practical machine learning tools and techniques (3rd ed.). Burlington, MA: Morgan Kaufmann.

Appendices

Appendix A: Table of all collected Features

For reference, below is a table of all feature names, feature types and which dataset they are associated with.

Table 8: Table of collected features

Feature Name	Feature Type	Raw Data Derivative
USR_NAME	String	Input when collection started
TIME	Numeric	Other
SBLINK_PEAK_IR_AVG	Numeric	Standard Blink
SBLINK_PEAK_IR_MIN	Numeric	Standard Blink
SBLINK_PEAK_IR_MAX	Numeric	Standard Blink
SBLINK_PEAK_IR_STDEV	Numeric	Standard Blink
SBLINK_PEAK_IR_VAR	Numeric	Standard Blink
SBLINK_PEAK_IR_MED	Numeric	Standard Blink
SBLINK_TO_CLOSE_TIME_AVG	Numeric	Standard Blink
SBLINK_ TO_CLOSE_TIME _MIN	Numeric	Standard Blink
SBLINK_ TO_CLOSE_TIME _MAX	Numeric	Standard Blink
SBLINK_ TO_CLOSE_TIME _STDEV	Numeric	Standard Blink
SBLINK_ TO_CLOSE_TIME _VAR	Numeric	Standard Blink
SBLINK_ TO_CLOSE_TIME _MED	Numeric	Standard Blink
SBLINK_TO_OPEN_TIME_AVG	Numeric	Standard Blink
SBLINK_ TO_ OPEN _TIME _MIN	Numeric	Standard Blink
SBLINK_ TO_ OPEN _TIME _MAX	Numeric	Standard Blink
SBLINK_ TO_ OPEN _TIME _STDEV	Numeric	Standard Blink
SBLINK_ TO_ OPEN _TIME _VAR	Numeric	Standard Blink
SBLINK_ TO_ OPEN _TIME _MED	Numeric	Standard Blink
SBLINK_BETWEEN_TIME_AVG	Numeric	Standard Blink
SBLINK_ BETWEEN _TIME _MIN	Numeric	Standard Blink
SBLINK_ BETWEEN _TIME _MAX	Numeric	Standard Blink

SBLINK_ BETWEEN _TIME _STDEV	Numeric	Standard Blink
SBLINK_ BETWEEN _TIME _VAR	Numeric	Standard Blink
SBLINK_ BETWEEN _TIME _MED	Numeric	Standard Blink
SBLINK_MIN_IR_AVG	Numeric	Standard Blink
SBLINK_ MIN_IR_MED	Numeric	Standard Blink
SBLINK_ MIN _IR_MAX	Numeric	Standard Blink
SBLINK_ MIN _IR_STDEV	Numeric	Standard Blink
SBLINK_ MIN _IR_VAR	Numeric	Standard Blink
SBLINK_SUM_PEAKS	Numeric	Standard Blink
SBLINK _IR_AVG	Numeric	Standard Blink
SBLINK_IR_MIN	Numeric	Standard Blink
SBLINK_IR_MAX	Numeric	Standard Blink
SBLINK_IR_STDEV	Numeric	Standard Blink
SBLINK_IR_VAR	Numeric	Standard Blink
SBLINK_IR_MED	Numeric	Standard Blink
SIBLINK_PEAK_IR_AVG	Numeric	Induced + Standard Blink
SIBLINK_PEAK_IR_MIN	Numeric	Induced + Standard Blink
SIBLINK_PEAK_IR_MAX	Numeric	Induced + Standard Blink
SIBLINK_PEAK_IR_STDEV	Numeric	Induced + Standard Blink
SIBLINK_PEAK_IR_VAR	Numeric	Induced + Standard Blink
SIBLINK_PEAK_IR_MED	Numeric	Induced + Standard Blink
SIBLINK_TO_CLOSE_TIME_AVG	Numeric	Induced + Standard Blink
SIBLINK_ TO_CLOSE_TIME _MIN	Numeric	Induced + Standard Blink
SIBLINK_ TO_CLOSE_TIME _MAX	Numeric	Induced + Standard Blink
SIBLINK_ TO_CLOSE_TIME _STDEV	Numeric	Induced + Standard Blink
SIBLINK_ TO_CLOSE_TIME _VAR	Numeric	Induced + Standard Blink
SIBLINK_ TO_CLOSE_TIME _MED	Numeric	Induced + Standard Blink
SIBLINK_TO_OPEN_TIME_AVG	Numeric	Induced + Standard Blink
SIBLINK_ TO_ OPEN _TIME _MIN	Numeric	Induced + Standard Blink
SIBLINK_ TO_ OPEN _TIME _MAX	Numeric	Induced + Standard Blink
SIBLINK_ TO_ OPEN _TIME _STDEV	Numeric	Induced + Standard Blink

SIBLINK_ TO_ OPEN _TIME _VAR	Numeric	Induced + Standard Blink
SIBLINK_ TO_ OPEN _TIME _MED	Numeric	Induced + Standard Blink
SIBLINK_BETWEEN_TIME_AVG	Numeric	Induced + Standard Blink
SIBLINK_ BETWEEN _TIME _MIN	Numeric	Induced + Standard Blink
SIBLINK_ BETWEEN _TIME _MAX	Numeric	Induced + Standard Blink
SIBLINK_ BETWEEN _TIME _STDEV	Numeric	Induced + Standard Blink
SIBLINK_ BETWEEN _TIME _VAR	Numeric	Induced + Standard Blink
SIBLINK_ BETWEEN _TIME _MED	Numeric	Induced + Standard Blink
SIBLINK_MIN_IR_AVG	Numeric	Induced + Standard Blink
SIBLINK_ MIN_IR_MED	Numeric	Induced + Standard Blink
SIBLINK_ MIN _IR_MAX	Numeric	Induced + Standard Blink
SIBLINK_ MIN _IR_STDEV	Numeric	Induced + Standard Blink
SIBLINK_ MIN _IR_VAR	Numeric	Induced + Standard Blink
SIIBLINK_SUM_PEAKS	Numeric	Induced + Standard Blink
SIBLINK _IR_AVG	Numeric	Induced + Standard Blink
SIBLINK_IR_MIN	Numeric	Induced + Standard Blink
SIBLINK_IR_MAX	Numeric	Induced + Standard Blink
SIBLINK_IR_STDEV	Numeric	Induced + Standard Blink
SIBLINK_IR_VAR	Numeric	Induced + Standard Blink
SIBLINK_IR_MED	Numeric	Induced + Standard Blink
TOTAL_SHM_X	Numeric	Gyroscope
PEAK_SHM_X_AVG	Numeric	Gyroscope
PEAK_SHM_X_MIN	Numeric	Gyroscope
PEAK_SHM_X_MAX	Numeric	Gyroscope
PEAK_SHM_X_STDEV	Numeric	Gyroscope
PEAK_SHM_X_VAR	Numeric	Gyroscope
PEAK_SHM_X_MED	Numeric	Gyroscope
TIME_SHM_X_MIN	Numeric	Gyroscope
TIME_SHM_X_MAX	Numeric	Gyroscope
SHM_X_AVG	Numeric	Gyroscope
SHM_X_MIN	Numeric	Gyroscope

SHM_X_MAX	Numeric	Gyroscope
SHM_X_STDEV	Numeric	Gyroscope
SHM_X_VAR	Numeric	Gyroscope
SHM_X_MED	Numeric	Gyroscope
TOTAL_SHM_Y	Numeric	Gyroscope
PEAK_SHM_Y_AVG	Numeric	Gyroscope
PEAK_SHM_Y_MIN	Numeric	Gyroscope
PEAK_SHM_Y_MAX	Numeric	Gyroscope
PEAK_SHM_Y_STDEV	Numeric	Gyroscope
PEAK_SHM_Y_VAR	Numeric	Gyroscope
PEAK_SHM_Y_MED	Numeric	Gyroscope
TIME_SHM_Y_MIN	Numeric	Gyroscope
TIME_SHM_Y_MAX	Numeric	Gyroscope
SHM_Y_AVG	Numeric	Gyroscope
SHM_Y_MIN	Numeric	Gyroscope
SHM_Y_MAX	Numeric	Gyroscope
SHM_Y_STDEV	Numeric	Gyroscope
SHM_Y_VAR	Numeric	Gyroscope
SHM_Y_MED	Numeric	Gyroscope
TOTAL_SHM_Z	Numeric	Gyroscope
PEAK_SHM_Z_AVG	Numeric	Gyroscope
PEAK_SHM_Z_MIN	Numeric	Gyroscope
PEAK_SHM_Z_MAX	Numeric	Gyroscope
PEAK_SHM_Z_STDEV	Numeric	Gyroscope
PEAK_SHM_Z_VAR	Numeric	Gyroscope
PEAK_SHM_Z_MED	Numeric	Gyroscope
TIME_SHM_Z_MIN	Numeric	Gyroscope
TIME_SHM_Z_MAX	Numeric	Gyroscope
SHM_Z_AVG	Numeric	Gyroscope
SHM_Z_MIN	Numeric	Gyroscope
SHM_Z_MAX	Numeric	Gyroscope

SHM_Z_STDEV	Numeric	Gyroscope
SHM_Z_VAR	Numeric	Gyroscope
SHM_Z_MED	Numeric	Gyroscope
TOTAL_SHM_X	Numeric	Accelerometer
PEAK_HM_X_AVG	Numeric	Accelerometer
PEAK_HM_X_MIN	Numeric	Accelerometer
PEAK_HM_X_MAX	Numeric	Accelerometer
PEAK_HM_X_STDEV	Numeric	Accelerometer
PEAK_HM_X_VAR	Numeric	Accelerometer
PEAK_HM_X_MED	Numeric	Accelerometer
TIME_HM_X_MIN	Numeric	Accelerometer
TIME_HM_X_MAX	Numeric	Accelerometer
HM_X_AVG	Numeric	Accelerometer
HM_X_MIN	Numeric	Accelerometer
HM_X_MAX	Numeric	Accelerometer
HM_X_STDEV	Numeric	Accelerometer
SHM_X_VAR	Numeric	Accelerometer
HM_X_MED	Numeric	Accelerometer
TOTAL_HM_Y	Numeric	Accelerometer
PEAK_HM_Y_AVG	Numeric	Accelerometer
PEAK_HM_Y_MIN	Numeric	Accelerometer
PEAK_HM_Y_MAX	Numeric	Accelerometer
PEAK_HM_Y_STDEV	Numeric	Accelerometer
PEAK_HM_Y_VAR	Numeric	Accelerometer
PEAK_HM_Y_MED	Numeric	Accelerometer
TIME_HM_Y_MIN	Numeric	Accelerometer
TIME_HM_Y_MAX	Numeric	Accelerometer
HM_Y_AVG	Numeric	Accelerometer
HM_Y_MIN	Numeric	Accelerometer
HM_Y_MAX	Numeric	Accelerometer
HM_Y_STDEV	Numeric	Accelerometer

HM_Y_VAR	Numeric	Accelerometer
HM_Y_MED	Numeric	Accelerometer
TOTAL_HM_Z	Numeric	Accelerometer
PEAK_HM_Z_AVG	Numeric	Accelerometer
PEAK_HM_Z_MIN	Numeric	Accelerometer
PEAK_HM_Z_MAX	Numeric	Accelerometer
PEAK_HM_Z_STDEV	Numeric	Accelerometer
PEAK_HM_Z_VAR	Numeric	Accelerometer
PEAK_HM_Z_MED	Numeric	Accelerometer
TIME_HM_Z_MIN	Numeric	Accelerometer
TIME_HM_Z_MAX	Numeric	Accelerometer
HM_Z_AVG	Numeric	Accelerometer
HM_Z_MIN	Numeric	Accelerometer
HM_Z_MAX	Numeric	Accelerometer
HM_Z_STDEV	Numeric	Accelerometer
HM_Z_VAR	Numeric	Accelerometer
HM_Z_MED	Numeric	Accelerometer

Appendix B: Relevant Features Table

All features are listed in order of relevancy to Functional Trees. Type is also indicated.

Table 9: Relevant features with rank

Rank	Feature	Type
0.698	147 SBLINK_MIN_IR_AVG	SB
0.694	119 SIBLINK_IR_AVG	IB
0.671	109 SIBLINK_MIN_IR_AVG	IB
0.664	102 SIBLINK_IR_MED	IB
0.662	151 SBLINK_MIN_IR_MAX	SB
0.66	92 SIBLINK_MIN_IR_MED	SB
0.657	161 SBLINK_IR_MIN	SB
0.655	130 SBLINK_IR_MED	IB
0.655	134 SBLINK_MIN_IR_MED	SB
0.655	157 SBLINK_PEAK_IR_MIN	SB
0.652	141 SBLINK_IR_AVG	SB
0.651	124 SIBLINK_MIN_IR_MAX	IB
0.646	154 SBLINK_PEAK_IR_AVG	SB
0.64	90 PEAK_HM_Z_MIN	HM
0.636	128 SBLINK_IR_MAX	IB
0.633	101 SIBLINK_IR_MAX	IB
0.632	95 SIBLINK_PEAK_IR_MAX	IB
0.631	24 PEAK_HM_Z_AVG	HM
0.629	156 SBLINK_PEAK_IR_MED	SB
0.627	94 SIBLINK_PEAK_IR_MED	IB
0.626	112 SIBLINK_IR_MIN	IB
0.626	133 SBLINK_PEAK_IR_MAX	SB
0.624	108 SIBLINK_PEAK_IR_MIN	IB
0.622	88 PEAK_HM_Z_STDEV	HM
0.613	99 SIBLINK_PEAK_IR_AVG	IB
0.608	41 PEAK_HM_Z_MAX	HM

0.601	73 HM_Z_MAX	HM
0.59	35 PEAK_HM_Z_VAR	HM
0.583	80 HM_Z_MED	HM
0.561	71 SHM_Y_STDEV	SHM
0.547	8 HM_X_AVG	HM
0.544	21 HM_Z_AVG	HM
0.538	46 SHM_Y_VAR	SHM
0.533	39 PEAK_SHM_Y_MIN	SHM
0.529	79 PEAK_HM_Y_AVG	HM
0.515	27 SHM_Z_MIN	SHM
0.51	2 PEAK_HM_Z_MED	HM
0.509	74 SHM_Y_AVG	SHM
0.503	11 PEAK_SHM_Y_AVG	SHM
0.498	53 HM_X_MED	HM
0.494	52 HM_X_MIN	HM
0.494	76 PEAK_HM_Y_MIN	HM
0.483	34 SHM_Z_STDEV	SHM
0.48	12 HM_Y_MAX	HM
0.479	87 HM_Y_AVG	HM
0.476	84 SHM_Z_AVG	SHM
0.475	1 PEAK_HM_Y_MAX	HM
0.467	31 SHM_Y_MAX	SHM
0.467	59 HM_Z_MIN	HM
0.462	5 HM_X_VAR	HM
0.462	14 HM_X_STDEV	HM
0.458	93 SIBLINK_IR_VAR	IB
0.458	96 SIBLINK_IR_STDEV	IB
0.451	45 SHM_X_STDEV	SHM
0.45	78 PEAK_SHM_X_AVG	SHM
0.443	50 PEAK_HM_X_STDEV	HM
0.438	23 PEAK_SHM_X_MIN	SHM

0.436	69 PEAK_SHM_Y_MAX	SHM
0.431	118 SIBLINK_MIN_IR_STDEV	IB
0.431	127 SIBLINK_MIN_IR_VAR	IB
0.424	15 SHM_Y_MIN	SHM
0.422	26 HM_Z_STDEV	HM
0.422	82 HM_Z_VAR	HM
0.414	89 PEAK_SHM_Z_MIN	SHM
0.413	3 PEAK_HM_X_MIN	HM
0.411	37 PEAK_SHM_Z_AVG	SHM
0.411	70 SHM_X_MAX	SHM
0.409	60 PEAK_SHM_Y_STDEV	SHM
0.407	4 PEAK_SHM_X_STDEV	SHM
0.404	33 HM_Y_STDEV	HM
0.404	68 HM_Y_VAR	HM
0.404	77 SHM_X_MIN	SHM
0.401	30 PEAK_SHM_Z_MAX	SHM
0.401	51 PEAK_SHM_X_MAX	SHM
0.399	64 SHM_X_VAR	SHM
0.397	6 HM_Y_MIN	HM
0.393	29 PEAK_HM_X_AVG	HM
0.39	86 HM_X_MAX	HM
0.39	132 SBLINK_MIN_IR_STDEV	SB
0.39	144 SBLINK_MIN_IR_VAR	SB
0.365	58 SHM_Z_MAX	SHM
0.357	55 PEAK_HM_X_MAX	HM
0.354	47 SHM_Z_VAR	SHM
0.351	16 PEAK_HM_Y_STDEV	HM
0.347	44 PEAK_HM_Y_VAR	HM
0.34	18 PEAK_SHM_Z_STDEV	SHM
0.338	131 SBLINK_IR_VAR	SB
0.338	136 SBLINK_IR_STDEV	SB

0.33	17 PEAK_HM_X_VAR	HM
0.317	85 PEAK_SHM_Y_VAR	SHM
0.312	32 PEAK_HM_Y_MED	HM
0.306	110 SIBLINK_TO_CLOSE_TIME_MED	IB
0.3	107 SIBLINK_SUM_PEAKS	IB
0.289	28 PEAK_SHM_X_VAR	SHM
0.265	65 TOTAL_SHM_Y	SHM
0.261	103 SIBLINK_PEAK_IR_STDEV	IB
0.261	122 SIBLINK_PEAK_IR_VAR	IB
0.258	61 TOTAL_SHM_X	SHM
0.256	106 SIBLINK_BETWEEN_TIME_VAR	IB
0.254	25 PEAK_SHM_Z_VAR	SHM
0.245	75 TOTAL_HM_X	HM
0.23	7 SHM_Z_MED	SHM
0.227	116 SIBLINK_TO_CLOSE_TIME_MAX	IB
0.216	139 SBLINK_SUM_PEAKS	SB
0.215	137 SBLINK_PEAK_IR_VAR	SB
0.215	158 SBLINK_PEAK_IR_STDEV	SB
0.212	81 SHM_X_AVG	SHM
0.202	120 SIBLINK_TO_CLOSE_TIME_AVG	IB

Appendix C: Consent Form

Informed Consent Agreement for Participation in a Research Study

Investigator: Krishna Venkatasubramanian, PhD
Contact Information: Fuller Lab 137, Computer Science Department, kven@wpi.edu., 508-831-6571
Title of Research Study: BlinkPrint: Using blinking patterns for authentication purposes on head mounted wearable technology.

Introduction (recommended)
You are being asked to participate in a research study. Before you agree, however, you must be fully informed about the purpose of the study, the procedures to be followed, and any benefits, risks or discomfort that you may experience as a result of your participation. This form presents information about the study so that you may make a fully informed decision regarding your participation.

Purpose of the study: The purpose of the study is to determine if a person's blinking is unique enough to uniquely identify them. If this hypothesis holds then the resultant technology will be like developing a password system for wearable technologies such as Google Glass, where the password entry will simply be the subject blinking.

Procedures to be followed: In order to determine if blinking can be used to authenticate (unique identify) a person we need to collect some data regarding a person's blinking pattern. We will ask you to sit in a comfortable position, wearable Google Glass device on your eyes and then watch a set of numbers and alphabets "cue-cards" being displayed on your Google Glass screen and look for specific numbers and alphabets. The purpose of the "cue-cards" is simply to keep you focused for the duration of the data collection. The entire process should take about 20 minutes. During this entire process all you are expected to do is watch the "cue-cards" and blink normally. We plan to record the following information during this data collection process:

- Your blinking
- Your head movements
- Your gender
- Your age
- Whether you are wearing contacts lenses and whether you wear glasses

If you wear glasses please remove them before putting on the Google Glass. If you were contact lenses you can leave them in.

Risks to study participants: We do not anticipate any risks or adverse events wearing the Google Glass for such a short period of time. If however you are feeling uncomfortable at any stage of the data collection, please STOP IMMEDIATELY. At that time if you do not wish to continue, all your data will be erased immediately.

Benefits to research participants and others: Head mounted wearable technology will become more and more prevalent in the near future. They will be useful in a variety of applications from surveillance to services in public places like airports etc. It is therefore important to ensure that the person wearing such technology is trustworthy. Being able to to authenticate the users of such technology is the first step in ensuring this property.

Record keeping and confidentiality The data will be collected by the students performing the MQP and stored in the laptop provided to them by the PI. The data will be stored on the PI provided laptop and on the server of the CS department. Both the laptop and the server account where the data is stored are password protected and will be accessible only to the PI. The students performing the MQP will be able to access the data during C-term 2015 while they finish their MQP.

Compensation or treatment in the event of injury: We do not anticipate any injury during the course of our data collection.

For more information about this research or about the rights of research participants, or in case of research-related injury, contact: (Fill in your contact information or make reference to information provided at top of page. In addition, include the contact information for the IRB Chair (Professor Kent Rissmiller, Tel. 508-831-5019, Email: kjr@wpi.edu) and the University Compliance Officer (Michael J. Curley, Tel. 508-831-6919, Email: mjcurley@wpi.edu). This section is required.)

Your participation in this research is voluntary. Your refusal to participate will not result in any penalty to you or any loss of benefits to which you may otherwise be entitled. You may decide to stop participating in the research at any time without penalty or loss of other benefits. The project investigators retain the right to cancel or postpone the experimental procedures at any time they see fit.

By signing below, you acknowledge that you have been informed about and consent to be a participant in the study described above. Make sure that your questions are answered to your satisfaction before signing. You are entitled to retain a copy of this consent agreement.

_____ Date: _____
Study Participant Signature

_____
Study Participant Name (Please print)

_____ Date: _____
Signature of Person who explained this study

WPI IRB 1
APPROVED
2/20/15-2/19/15

Appendix D: Background Information Recorded from Test Subjects

Original Investigators: Alexander W. Witt, Alexander D. Solomon, Cynthia E. Rogers
Date Produced: 02/24/2015

UserOne

Investigator Performing Study	=	Alexander D. Solomon
Date of Study Participation	=	Wednesday, Feb. 11, 2015
Test Subject is WPI Student	=	Yes
Test Subject Provided Age	=	21
Test Subject Provided Gender	=	Male
Test Subject Typically Wears Glasses	=	Yes
Test Subject is Wearing Contacts for Study	=	Yes
Location of Performed Study	=	Off Campus Apartment
Start Time of Performed Study	=	2:00 PM

UserTwo

Investigator Performing Study	=	Cynthia E. Rogers
Date of Study Participation	=	Tuesday, Feb. 10, 2015
Test Subject is WPI Student	=	Yes
Test Subject Provided Age	=	21
Test Subject Provided Gender	=	Female
Test Subject Typically Wears Glasses	=	Yes
Test Subject is Wearing Contacts for Study	=	No
Location of Performed Study	=	Library
Start Time of Performed Study	=	8:00 PM

UserThree

Investigator Performing Study	=	Alexander W. Witt
Date of Study Participation	=	Tuesday, Feb. 10, 2015
Test Subject is WPI Student	=	Yes
Test Subject Provided Age	=	21
Test Subject Provided Gender	=	Male
Test Subject Typically Wears Glasses	=	Yes
Test Subject is Wearing Contacts for Study	=	No
Location of Performed Study	=	Isolated Cubicle of WPI Gordon C. Library
Start Time of Performed Study	=	9:00 PM

UserFour

Investigator Performing Study	=	Cynthia E. Rogers
Date of Study Participation	=	Wednesday, Feb. 25, 2015
Test Subject is WPI Student	=	Yes
Test Subject Provided Age	=	22
Test Subject Provided Gender	=	Male
Test Subject Typically Wears Glasses	=	No
Test Subject is Wearing Contacts for Study	=	No
Location of Performed Study	=	Off-Campus
Start Time of Performed Study	=	10:05 AM

UserFive

Investigator Performing Study	=	Alexander W. Witt
Date of Study Participation	=	Sunday, Feb. 22, 2015
Test Subject is WPI Student	=	Yes
Test Subject Provided Age	=	18
Test Subject Provided Gender	=	Male
Test Subject Typically Wears Glasses	=	Yes
Test Subject is Wearing Contacts for Study	=	No
Location of Performed Study	=	Zoo Lab of WPI Fuller Labs Building
Start Time of Performed Study	=	4:15 PM

UserSix

Investigator Performing Study	=	Alexander W. Witt
Date of Study Participation	=	Monday, Feb. 23, 2015
Test Subject is WPI Student	=	Yes
Test Subject Provided Age	=	25
Test Subject Provided Gender	=	Male
Test Subject Typically Wears Glasses	=	Yes
Test Subject is Wearing Contacts for Study	=	No
Location of Performed Study	=	Zoo Lab of WPI Fuller Labs Building
Start Time of Performed Study	=	4:05 PM

UserSeven

Investigator Performing Study	=	Alexander D. Solomon
Date of Study Participation	=	Monday, Feb. 23, 2015
Test Subject is WPI Student	=	Yes
Test Subject Provided Age	=	20
Test Subject Provided Gender	=	Female
Test Subject Typically Wears Glasses	=	No
Test Subject is Wearing Contacts for Study	=	No
Location of Performed Study	=	Third Floor of Campus Center
Start Time of Performed Study	=	9:00 PM

UserEight

Investigator Performing Study	=	Alexander D. Solomon
Date of Study Participation	=	Monday, Feb. 23, 2015
Test Subject is WPI Student	=	Yes
Test Subject Provided Age	=	21
Test Subject Provided Gender	=	Female
Test Subject Typically Wears Glasses	=	No
Test Subject is Wearing Contacts for Study	=	No
Location of Performed Study	=	Third Floor of Campus Center
Start Time of Performed Study	=	9:30 PM

UserNine

Investigator Performing Study	=	Alexander D. Solomon
Date of Study Participation	=	Thursday, Feb. 26, 2015
Test Subject is WPI Student	=	Yes
Test Subject Provided Age	=	21
Test Subject Provided Gender	=	Male
Test Subject Typically Wears Glasses	=	No
Test Subject is Wearing Contacts for Study	=	No
Location of Performed Study	=	Off Campus Apartment
Start Time of Performed Study	=	9:00 PM

UserTen

Investigator Performing Study	=	Alexander D. Solomon
Date of Study Participation	=	Thursday, Feb. 26, 2015
Test Subject is WPI Student	=	Yes
Test Subject Provided Age	=	20
Test Subject Provided Gender	=	Female
Test Subject Typically Wears Glasses	=	Yes, still able to see screen clearly without
Test Subject is Wearing Contacts for Study	=	No
Location of Performed Study	=	Off Campus Apartment
Start Time of Performed Study	=	11:00 PM

UserEleven

Investigator Performing Study	=	Cynthia E. Rogers
Date of Study Participation	=	Wednesday, Feb. 25, 2015
Test Subject is WPI Student	=	Yes
Test Subject Provided Age	=	25
Test Subject Provided Gender	=	Male
Test Subject Typically Wears Glasses	=	Yes
Test Subject is Wearing Contacts for Study	=	No
Location of Performed Study	=	Library
Start Time of Performed Study	=	8:47 AM

UserTwelve

Investigator Performing Study	=	Cynthia E. Rogers
Date of Study Participation	=	Wednesday, Feb. 25, 2015
Test Subject is WPI Student	=	Yes
Test Subject Provided Age	=	21
Test Subject Provided Gender	=	Female
Test Subject Typically Wears Glasses	=	No
Test Subject is Wearing Contacts for Study	=	No
Location of Performed Study	=	Off Campus Apartment
Start Time of Performed Study	=	12:00 PM

UserThirteen

Investigator Performing Study	=	Alexander W. Witt
Date of Study Participation	=	Friday, Feb. 27, 2015
Test Subject is WPI Student	=	Yes
Test Subject Provided Age	=	21
Test Subject Provided Gender	=	Male
Test Subject Typically Wears Glasses	=	No, but has slight astigmatism
Test Subject is Wearing Contacts for Study	=	No
Location of Performed Study	=	Fuller Labs A22 Adjacent to Zoo Lab
Start Time of Performed Study	=	8:30 PM

Additional Notes: The subject mentioned difficulty adjusting the device to comfortably view slide content even after adjusting the nosepiece and viewport of the Google Glass device to meet his own preferences. The subject also swiveled slightly in his chair.

Appendix E: WEKA and Data Mining

In our project, we collect a large amount of data and run machine learning algorithms on this data. We do this to find patterns in the data that will allow us to uniquely identify a user. Finding patterns in a large amount of data is the definition of data mining [21]. As computer scientists, our job is to understand how to use ML algorithms, rather than to create them. In our project, we choose to use WEKA to explore ML.

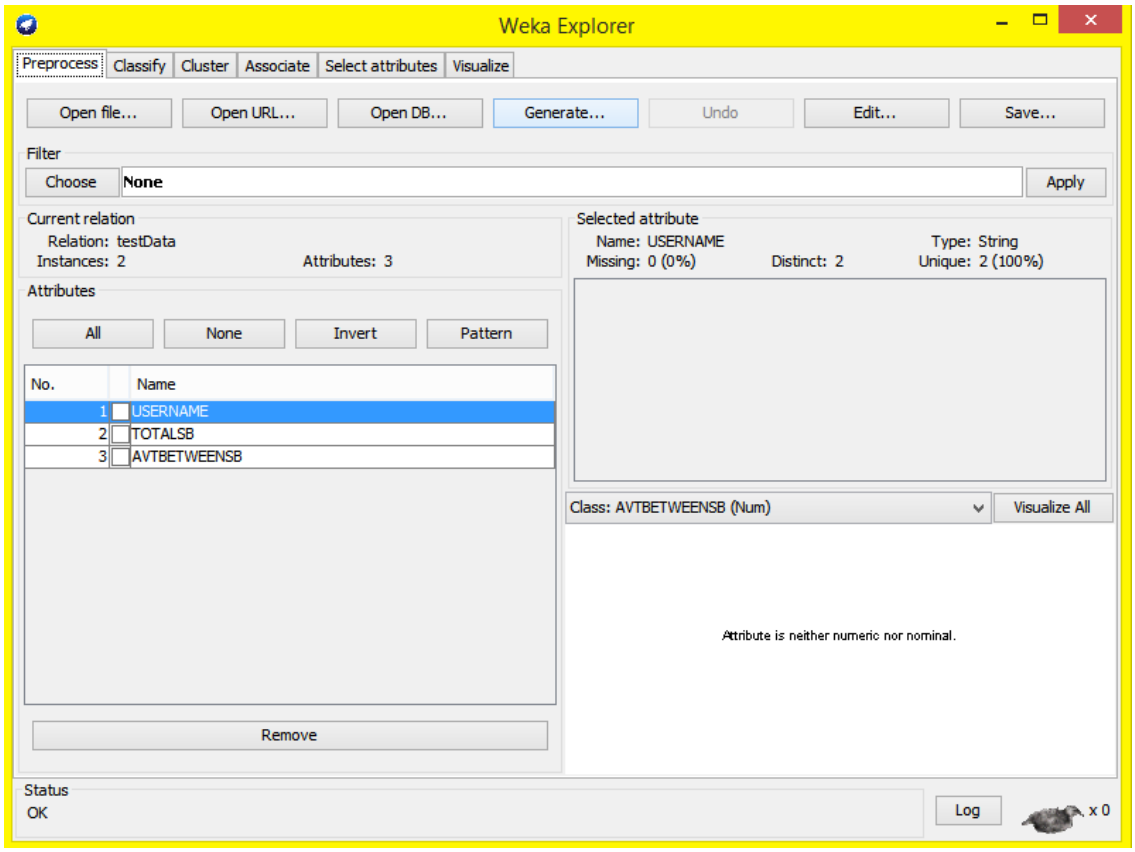

Figure 29: WEKA Explorer Preprocessing Screen

WEKA takes data in the form of an arff file. Below is an example of an arff file:

```
@relation testData

@attribute USERNAME STRING
@attribute TOTALSB numeric
@attribute AVTBETWEENSB numeric

@data
Joe, 1, 300
Bill, 1, 500
```

Arff files have 3 parts. The first is the r**elation** which defines the relation represented in the data. You can put any word in the relation as long as it's prudent to represent the file. Second, each feature you wish to be represented in the data is defined in the **attribute** section. These attributes (which map to our features) can be different data types (string, real, numeric or deterministic). Third, the **data** field will collect the data that will be input into the scenario. In our report, each line in the data section of the arff file is equivalent to one collection from the HMD. What this means is that we have one running arff file with many lines of data. Once we've created this file, we open it in WEKA explorer and explore it using different ML algorithms.

While data mining, it is a good practice to allow the algorithm to decide the usefulness of each feature (feature selection). The figure below depicts the Select Attributes window in WEKA which allows us to select features. The Select Attribute tab is depicted in Figure 30.

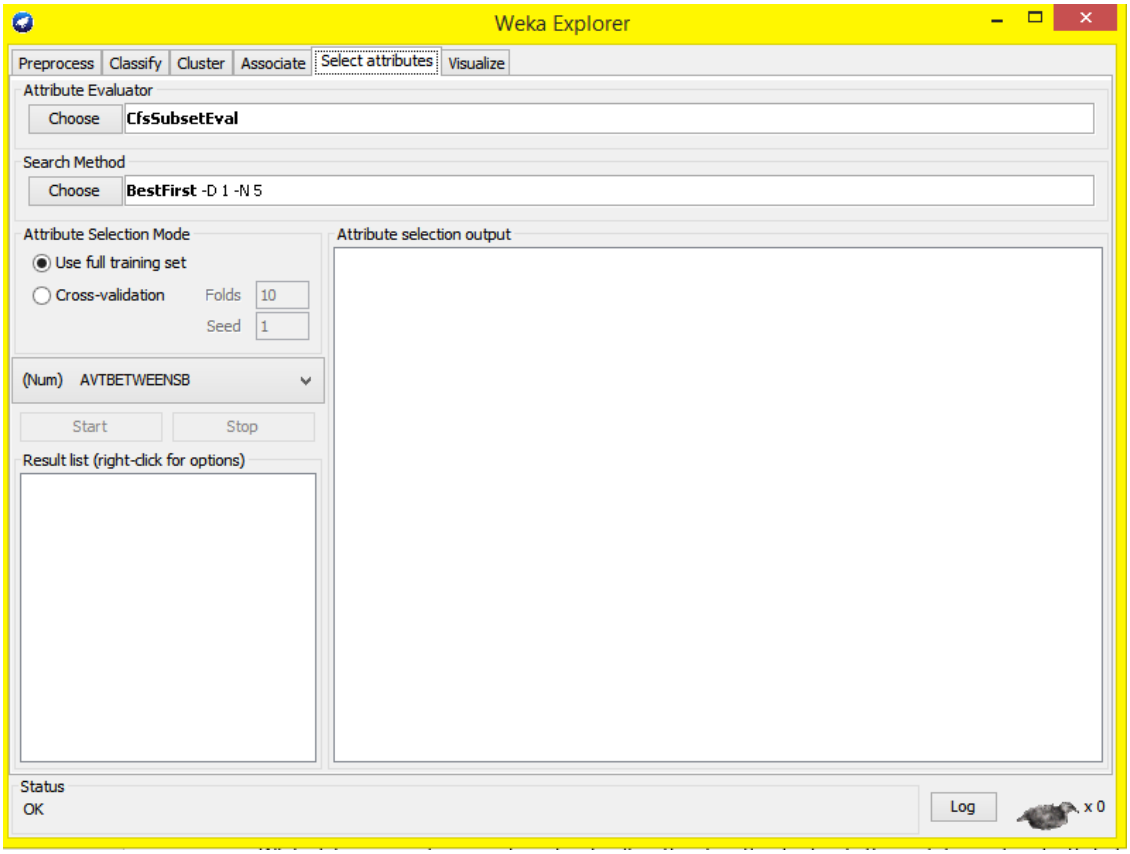

Figure 30: Select attribute tab in WEKA

Once we've selected the best features and determined which algorithm works the best, we use the "Classify" tab to generate the performance analysis of an algorithm. Below is the classification pane of WEKA. The classifier tab is seen in Figure 31.

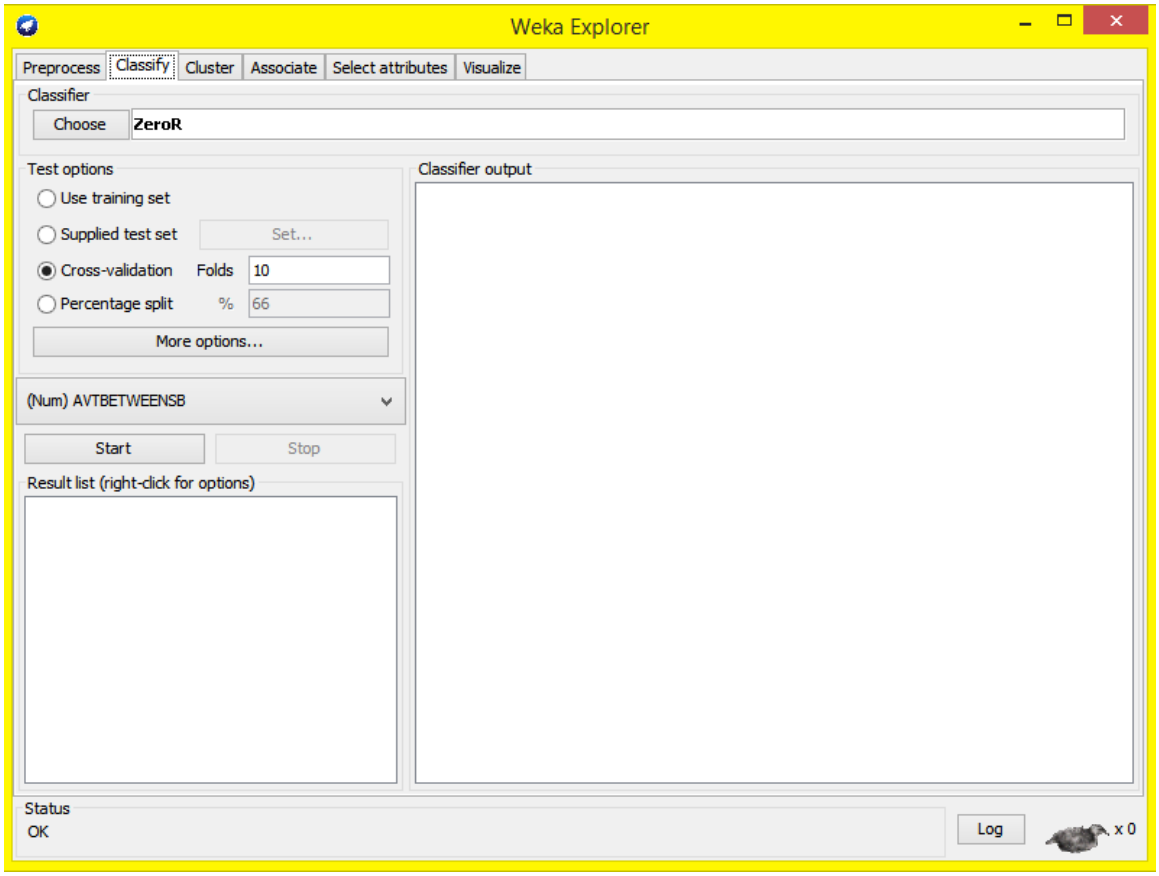

Figure 31: Classifier tab in WEKA

Appendix F: Step by Experimental Methodology

Formal Testing Procedure for the Experimenter

Original Investigators: Alexander W. Witt, Alexander D. Solomon, Cynthia E. Rogers

Date Produced: 02/24/2015

Prior to Performing Testing

1) Ensure that the Google Glass device is sufficiently charged
2) Ensure that the development computer is sufficiently charged
3) Allocate a space that is sufficiently quiet and that uses fluorescent lighting

Preparing the Testing Apparatus

1) Connect the Google Glass device to the development computer using the USB cable
2) Turn on the Google Glass device using the button that is on the right inner-face
3) Turn on the development computer and login to the appropriate domain
4) Once the laptop is booted and you have logged in, go to the Windows 7 Start orb
5) Use the Start orb to search for the Command Prompt utility
6) Open the Command Prompt utility
7) Type the following into the command line: > adb root
8) Hit the return key
9) Type the following into the command line: > adb shell
10) Hit the return key
11) Type the following into the command line: > chmod 777 /sys/bus/i2c/devices/4-0035/proxraw
12) Hit the return key
13) Type the following into the command line: > exit
14) Close the Command Prompt utility
15) Once the Command Prompt utility has been closed, go back to the Windows 7 Start orb
16) Use the Start orb to search for the Eclipse IDE installed on the development computer
17) Click OK because the correct workspace is selected automatically by default
18) In the Eclipse "Package Explorer" tab navigate to Blink > src > com.davidkeeley.android.blinktothefuture.main and open the BlinkMain.java file
19) In the monitorThread of the windowFocusChanged() method, change the second parameter passed into monitor.onSlide() to be the username that you will associate with a given test subject
20) Save the change to the BlinkMain.java file
21) Close all the text file that is used for recording transferred data

Prepare the Test Subject for Testing

1) Read each section of the IRB form to the test subject and explain any ambiguity
2) If the test subject agrees to participate, ask them their age and whether or not they wear glasses and are currently wearing contacts
3) Record the answers provided from the test subject
4) Inform the test subject to notify you immediately if the application fails

5) Inform the test subject to notify you when a slide reading END appears in the application
6) Tell the test subject to adjust the eye piece and Google Glass device on their head until they are comfortable
7) Tell the user not to talk during the collection of data and only focus on the cue cards that they will be shown
8) Tell the user when you will be starting the application

Performing Testing on a Test Subject

1) Start the android application by selecting "Glass Application" under the run menu in Eclipse on the development computer
2) After hearing the sound triggered in the android application, start the transfer server on the development computer
3) If the transfer server does not start, but the console output instead indicates SOCKET CLOSED, attempt to start the server again
4) Wait until the test subject indicates that he / she has seen the END slide
5) Stop the transfer server
6) If the subject indicates eye strain, let the subject stop and then continue if they wish to continue
7) Repeat steps 1 through 6 two more times

Debriefing the Test Subject

1) Provide the subject with any additional information pertaining to how their data will be used.
2) Provide the user with an additional copy of the IRB form if they request such a copy

Removing False Starts

Description: Sometimes when the application fails to start when launching the server the username is printed without any data following it. In order to produce a proper ARFF, the experimenter should go through the most recent data in recorded in the text file that is used for recording all transferred data and should remove those additional printed lines.

Producing an ARFF File for Analysis of the Data in WEKA

1) After removing any false starts, save the file used for recording all transferred data
2) Close the file used for recording all transferred data
3) Go to the run menu in Eclipse
4) Select "testArffBuilder"
5) Run the ArffBuilder over the file used for recording all transferred data

Appendix G: Functional Tree Raw Data for Relevant Feature Sets

Full Feature Set, Relevant only.

User	Accuracy	Precision	Recall	F-Score	FP Rate	FN Rate	TP	FP	FN	TN
User 3	0.997436	1	0.966667	0.983051	0	0.033333	29	0	1	360
User2	0.982051	0.870968	0.9	0.885246	0.011111	0.1	27	4	3	356
User1	0.979487	0.84375	0.9	0.870968	0.013889	0.1	27	5	3	355
User 5	0.979487	0.923077	0.8	0.857143	0.005556	0.2	24	2	6	358
User 6	0.989744	0.933333	0.933333	0.933333	0.005556	0.066667	28	2	2	358
User 7	0.976923	0.83871	0.866667	0.852459	0.013889	0.133333	26	5	4	355
User 11	0.984615	0.9	0.9	0.9	0.008333	0.1	27	3	3	357
User4	0.984615	0.9	0.9	0.9	0.008333	0.1	27	3	3	357
User 8	0.969231	0.75	0.9	0.818182	0.025	0.1	27	9	3	351
User 9	0.974359	0.884615	0.766667	0.821429	0.008333	0.233333	23	3	7	357
User 10	0.964103	0.785714	0.733333	0.758621	0.016667	0.266667	22	6	8	354
User 12	0.997436	0.967742	1	0.983607	0.002778	0	30	1	0	359
User 13	0.994872	0.966667	0.966667	0.966667	0.002778	0.033333	29	1	1	359
Avg.	98%	89%	89%	89%	1%	11%				

Blink Feature Set

User	accuracy	Precision	Recall	F-Score	FP Rate	FN Rate	TP	FP	FN	TN
User 3	0.997436	0.967742	1	0.983607	0.002778	0	30	1	0	359
User2	0.930769	0.548387	0.566667	0.557377	0.038889	0.433333	17	14	13	346
User1	0.941026	0.606061	0.666667	0.634921	0.036111	0.333333	20	13	10	347
User 5	0.953846	0.6875	0.733333	0.709677	0.027778	0.266667	22	10	8	350
User 6	0.971795	0.851852	0.766667	0.807018	0.011111	0.233333	23	4	7	356
User 7	0.95641	0.685714	0.8	0.738462	0.030556	0.2	24	11	6	349
User 11	0.966667	0.774194	0.8	0.786885	0.019444	0.2	24	7	6	353
User4	0.938462	0.625	0.5	0.555556	0.025	0.5	15	9	15	351
User 8	0.930769	0.555556	0.5	0.526316	0.033333	0.5	15	12	15	348
User 9	0.920513	0.481481	0.433333	0.45614	0.038889	0.566667	13	14	17	346
User 10	0.925641	0.518519	0.466667	0.491228	0.036111	0.533333	14	13	16	347
User 12	0.976923	0.8	0.933333	0.861538	0.019444	0.066667	28	7	2	353
User 13	0.958974	0.733333	0.733333	0.733333	0.022222	0.266667	22	8	8	352
Avg.	95%	68%	68%	68%	3%	32%				

Induced Blink Feature Set

User	accuracy	Precision	Recall	F-Score	FP Rate	FN Rate	TP	FP	FN	TN
User 3	0.994872	0.9375	1	0.967742	0.005556	0	30	2	0	358
User2	0.948718	0.666667	0.666667	0.666667	0.027778	0.333333	20	10	10	350
User1	0.946154	0.628571	0.733333	0.676923	0.036111	0.266667	22	13	8	347
User 5	0.948718	0.647059	0.733333	0.6875	0.033333	0.266667	22	12	8	348
User 6	0.933333	0.566667	0.566667	0.566667	0.036111	0.433333	17	13	13	347
User 7	0.961538	0.741935	0.766667	0.754098	0.022222	0.233333	23	8	7	352
User 11	0.958974	0.694444	0.833333	0.757576	0.030556	0.166667	25	11	5	349
User4	0.930769	0.565217	0.433333	0.490566	0.027778	0.566667	13	10	17	350
User 8	0.920513	0.48	0.4	0.436364	0.036111	0.6	12	13	18	347
User 9	0.923077	0.5	0.366667	0.423077	0.030556	0.633333	11	11	19	349
User 10	0.928205	0.538462	0.466667	0.5	0.033333	0.533333	14	12	16	348
User 12	0.964103	0.75	0.8	0.774194	0.022222	0.2	24	8	6	352
User 13	0.964103	0.75	0.8	0.774194	0.022222	0.2	24	8	6	352
Avg.	95%	65%	66%	65%	3%	34%				

Standardized Blink Feature Set

User	accuracy	Precision	Recall	F-Score	FP Rate	FN Rate	TP	FP	FN	TN
User 3	0.979487	0.866667	0.866667	0.866667	0.011111	0.133333	26	4	4	356
User2	0.941026	0.62963	0.566667	0.596491	0.027778	0.433333	17	10	13	350
User1	0.928205	0.526316	0.666667	0.588235	0.05	0.333333	20	18	10	342
User 5	0.94359	0.625	0.666667	0.645161	0.033333	0.333333	20	12	10	348
User 6	0.958974	0.733333	0.733333	0.733333	0.022222	0.266667	22	8	8	352
User 7	0.941026	0.606061	0.666667	0.634921	0.036111	0.333333	20	13	10	347
User 11	0.94359	0.642857	0.6	0.62069	0.027778	0.4	18	10	12	350
User4	0.928205	0.541667	0.433333	0.481481	0.030556	0.566667	13	11	17	349
User 8	0.917949	0.461538	0.4	0.428571	0.038889	0.6	12	14	18	346
User 9	0.917949	0.464286	0.433333	0.448276	0.041667	0.566667	13	15	17	345
User 10	0.930769	0.548387	0.566667	0.557377	0.038889	0.433333	17	14	13	346
User 12	0.966667	0.774194	0.8	0.786885	0.019444	0.2	24	7	6	353
User 13	0.974359	0.8125	0.866667	0.83871	0.016667	0.133333	26	6	4	354
Avg.	94%	63%	64%	63%	3%	36%				

Movement Feature Set

User	accuracy	Precision	Recall	F-Score	FP Rate	FN Rate	TP	FP	FN	TN
User 3	0.984615	0.852941	0.966667	0.90625	0.013889	0.033333	29	5	1	355
User2	0.976923	0.888889	0.8	0.842105	0.008333	0.2	24	3	6	357
User1	0.946154	0.645161	0.666667	0.655738	0.030556	0.333333	20	11	10	349
User 5	0.94359	0.633333	0.633333	0.633333	0.030556	0.366667	19	11	11	349
User 6	0.953846	0.714286	0.666667	0.689655	0.022222	0.333333	20	8	10	352
User 7	0.95641	0.724138	0.7	0.711864	0.022222	0.3	21	8	9	352
User 11	0.969231	0.78125	0.833333	0.806452	0.019444	0.166667	25	7	5	353
User4	0.971795	0.827586	0.8	0.813559	0.013889	0.2	24	5	6	355
User 8	0.95641	0.709677	0.733333	0.721311	0.025	0.266667	22	9	8	351
User 9	0.953846	0.772727	0.566667	0.653846	0.013889	0.433333	17	5	13	355
User 10	0.94359	0.633333	0.633333	0.633333	0.030556	0.366667	19	11	11	349
User 12	0.992308	0.935484	0.966667	0.95082	0.005556	0.033333	29	2	1	358
User 13	0.979487	0.823529	0.933333	0.875	0.016667	0.066667	28	6	2	354
Avg.	96%	76%	76%	76%	2%	24%				

Gyroscope Feature Set

User	accuracy	Precision	Recall	F-Score	FP Rate	FN Rate	TP	FP	FN	TN
User 3	0.941026	0.612903	0.633333	0.622951	0.033333	0.366667	19	12	11	348
User2	0.935897	0.619048	0.433333	0.509804	0.022222	0.566667	13	8	17	352
User1	0.889744	0.30303	0.333333	0.31746	0.063889	0.666667	10	23	20	337
User 5	0.897436	0.352941	0.4	0.375	0.061111	0.6	12	22	18	338
User 6	0.915385	0.448276	0.433333	0.440678	0.044444	0.566667	13	16	17	344
User 7	0.930769	0.555556	0.5	0.526316	0.033333	0.5	15	12	15	348
User 11	0.925641	0.517241	0.5	0.508475	0.038889	0.5	15	14	15	346
User4	0.897436	0.333333	0.333333	0.333333	0.055556	0.666667	10	20	20	340
User 8	0.9	0.344828	0.333333	0.338983	0.052778	0.666667	10	19	20	341
User 9	0.928205	0.533333	0.533333	0.533333	0.038889	0.466667	16	14	14	346
User 10	0.884615	0.241379	0.233333	0.237288	0.061111	0.766667	7	22	23	338
User 12	0.958974	0.705882	0.8	0.75	0.027778	0.2	24	10	6	350
User 13	0.989744	0.90625	0.966667	0.935484	0.008333	0.033333	29	3	1	357
Avg.	92%	50%	49%	49%	4%	51%				

Accelerometer Feature Set

User	accuracy	Precision	Recall	F-Score	FP Rate	FN Rate	TP	FP	FN	TN
User 3	0.992308	0.965517	0.933333	0.949153	0.002778	0.066667	28	1	2	359
User2	0.961538	0.758621	0.733333	0.745763	0.019444	0.266667	22	7	8	353
User1	0.95641	0.782609	0.6	0.679245	0.013889	0.4	18	5	12	355
User 5	0.946154	0.636364	0.7	0.666667	0.033333	0.3	21	12	9	348
User 6	0.94359	0.642857	0.6	0.62069	0.027778	0.4	18	10	12	350
User 7	0.938462	0.615385	0.533333	0.571429	0.027778	0.466667	16	10	14	350
User 11	0.948718	0.647059	0.733333	0.6875	0.033333	0.266667	22	12	8	348
User4	0.961538	0.702703	0.866667	0.776119	0.030556	0.133333	26	11	4	349
User 8	1.417949	0.966184	6.666667	1.687764	0.019444	-5.66667	200	7	-170	353
User 9	0.948718	0.708333	0.566667	0.62963	0.019444	0.433333	17	7	13	353
User 10	0.948718	0.647059	0.733333	0.6875	0.033333	0.266667	22	12	8	348
User 12	0.989744	0.90625	0.966667	0.935484	0.008333	0.033333	29	3	1	357
User 13	0.925641	0.515152	0.566667	0.539683	0.044444	0.433333	17	16	13	344
Avg.	99%	73%	117%	78%	2%	-17%				

Appendix H: Multilayer Perceptron Raw Data for Relevant Features

Full Feature Set

User	accuracy	Precision	Recall	F-Score	FP Rate	FN Rate	TP	FP	FN	TN
User 3	0.994872	0.966667	0.966667	0.966667	0.002778	0.033333	29	1	1	359
User2	0.979487	0.84375	0.9	0.870968	0.013889	0.1	27	5	3	355
User1	0.966667	0.757576	0.833333	0.793651	0.022222	0.166667	25	8	5	352
User 5	0.984615	0.928571	0.866667	0.896552	0.005556	0.133333	26	2	4	358
User 6	0.989744	0.933333	0.933333	0.933333	0.005556	0.066667	28	2	2	358
User 7	0.976923	0.83871	0.866667	0.852459	0.013889	0.133333	26	5	4	355
User 11	0.976923	0.83871	0.866667	0.852459	0.013889	0.133333	26	5	4	355
User4	0.994872	0.966667	0.966667	0.966667	0.002778	0.033333	29	1	1	359
User 8	0.979487	0.866667	0.866667	0.866667	0.011111	0.133333	26	4	4	356
User 9	0.979487	0.892857	0.833333	0.862069	0.008333	0.166667	25	3	5	357
User 10	0.969231	0.846154	0.733333	0.785714	0.011111	0.266667	22	4	8	356
User 12	0.994872	0.966667	0.966667	0.966667	0.002778	0.033333	29	1	1	359
User 13	0.994872	0.966667	0.966667	0.966667	0.002778	0.033333	29	1	1	359
Avg.	98%	89%	89%	89%	1%	11%				

Blink Feature Set

User	accuracy	Precision	Recall	F-Score	FP Rate	FN Rate	TP	FP	FN	TN
User 3	0.997436	0.967742	1	0.983607	0.002778	0	30	1	0	359
User2	0.953846	0.730769	0.633333	0.678571	0.019444	0.366667	19	7	11	353
User1	0.95641	0.724138	0.7	0.711864	0.022222	0.3	21	8	9	352
User 5	0.930769	0.538462	0.7	0.608696	0.05	0.3	21	18	9	342
User 6	0.953846	0.730769	0.633333	0.678571	0.019444	0.366667	19	7	11	353
User 7	0.94359	0.617647	0.7	0.65625	0.036111	0.3	21	13	9	347
User 11	0.933333	0.576923	0.5	0.535714	0.030556	0.5	15	11	15	349
User4	0.928205	0.545455	0.4	0.461538	0.027778	0.6	12	10	18	350
User 8	0.928205	0.535714	0.5	0.517241	0.036111	0.5	15	13	15	347
User 9	0.915385	0.448276	0.433333	0.440678	0.044444	0.566667	13	16	17	344
User 10	0.925641	0.515152	0.566667	0.539683	0.044444	0.433333	17	16	13	344
User 12	0.94359	0.605263	0.766667	0.676471	0.041667	0.233333	23	15	7	345
User 13	0.976923	0.862069	0.833333	0.847458	0.011111	0.166667	25	4	5	356
Avg.	95%	65%	64%	64%	3%	36%				

Induced Blink Feature Set

User	accuracy	Precision	Recall	F-Score	FP Rate	FN Rate	TP	FP	FN	TN
User 3	0.994872	0.9375	1	0.967742	0.005556	0	30	2	0	358
User2	0.95641	0.933333	0.466667	0.622222	0.002778	0.533333	14	1	16	359
User1	0.946154	0.636364	0.7	0.666667	0.033333	0.3	21	12	9	348
User 5	0.95641	0.675676	0.833333	0.746269	0.033333	0.166667	25	12	5	348
User 6	0.948718	0.631579	0.8	0.705882	0.038889	0.2	24	14	6	346
User 7	0.95641	0.740741	0.666667	0.701754	0.019444	0.333333	20	7	10	353
User 11	0.928205	0.535714	0.5	0.517241	0.036111	0.5	15	13	15	347
User4	0.941026	0.684211	0.433333	0.530612	0.016667	0.566667	13	6	17	354
User 8	0.910256	0.407407	0.366667	0.385965	0.044444	0.633333	11	16	19	344
User 9	0.912821	0.433333	0.433333	0.433333	0.047222	0.566667	13	17	17	343
User 10	0.928205	0.53125	0.566667	0.548387	0.041667	0.433333	17	15	13	345
User 12	0.928205	0.53125	0.566667	0.548387	0.041667	0.433333	17	15	13	345
User 13	0.974359	0.75	1	0.857143	0.027778	0	30	10	0	350
Avg.	94%	65%	64%	63%	3%	36%				

Standardized Blink Feature Set

User	accuracy	Precision	Recall	F-Score	FP Rate	FN Rate	TP	FP	FN	TN
User 3	0.987179	0.878788	0.966667	0.920635	0.011111	0.033333	29	4	1	356
User2	0.935897	0.608696	0.466667	0.528302	0.025	0.533333	14	9	16	351
User1	0.910256	0.419355	0.433333	0.42623	0.05	0.566667	13	18	17	342
User 5	0.928205	0.538462	0.466667	0.5	0.033333	0.533333	14	12	16	348
User 6	0.951282	0.703704	0.633333	0.666667	0.022222	0.366667	19	8	11	352
User 7	0.941026	0.62069	0.6	0.610169	0.030556	0.4	18	11	12	349
User 11	0.923077	0.5	0.466667	0.482759	0.038889	0.533333	14	14	16	346
User4	0.925641	0.521739	0.4	0.45283	0.030556	0.6	12	11	18	349
User 8	0.884615	0.258065	0.266667	0.262295	0.063889	0.733333	8	23	22	337
User 9	0.917949	0.461538	0.4	0.428571	0.038889	0.6	12	14	18	346
User 10	0.887179	0.315789	0.4	0.352941	0.072222	0.6	12	26	18	334
User 12	0.923077	0.5	0.733333	0.594595	0.061111	0.266667	22	22	8	338
User 13	0.966667	0.757576	0.833333	0.793651	0.022222	0.166667	25	8	5	352
Avg.	93%	54%	54%	54%	4%	46%				

Movement Feature Set

User	accuracy	Precision	Recall	F-Score	FP Rate	FN Rate	TP	FP	FN	TN
User 3	0.989744	0.90625	0.966667	0.935484	0.008333	0.033333	29	3	1	357
User2	0.95641	0.709677	0.733333	0.721311	0.025	0.266667	22	9	8	351
User1	0.930769	0.56	0.466667	0.509091	0.030556	0.533333	14	11	16	349
User 5	0.923077	0.5	0.466667	0.482759	0.038889	0.533333	14	14	16	346
User 6	0.966667	0.793103	0.766667	0.779661	0.016667	0.233333	23	6	7	354
User 7	0.961538	0.777778	0.7	0.736842	0.016667	0.3	21	6	9	354
User 11	0.958974	0.71875	0.766667	0.741935	0.025	0.233333	23	9	7	351
User4	0.969231	0.75	0.9	0.818182	0.025	0.1	27	9	3	351
User 8	0.953846	0.714286	0.666667	0.689655	0.022222	0.333333	20	8	10	352
User 9	0.951282	0.761905	0.533333	0.627451	0.013889	0.466667	16	5	14	355
User 10	0.930769	0.540541	0.666667	0.597015	0.047222	0.333333	20	17	10	343
User 12	0.987179	0.903226	0.933333	0.918033	0.008333	0.066667	28	3	2	357
User 13	0.982051	0.848485	0.933333	0.888889	0.013889	0.066667	28	5	2	355
Avg.	96%	73%	73%	73%	2%	27%				

Gyroscope Feature Set

User	accuracy	Precision	Recall	F-Score	FP Rate	FN Rate	TP	FP	FN	TN
User 3	0.902564	0.4	0.533333	0.457143	0.066667	0.466667	16	24	14	336
User2	0.892308	0.2	0.133333	0.16	0.044444	0.866667	4	16	26	344
User1	0.884615	0.173913	0.133333	0.150943	0.052778	0.866667	4	19	26	341
User 5	0.915385	0.448276	0.433333	0.440678	0.044444	0.566667	13	16	17	344
User 6	0.897436	0.352941	0.4	0.375	0.061111	0.6	12	22	18	338
User 7	0.907692	0.428571	0.6	0.5	0.066667	0.4	18	24	12	336
User 11	0.910256	0.413793	0.4	0.40678	0.047222	0.6	12	17	18	343
User4	0.874359	0.27907	0.4	0.328767	0.086111	0.6	12	31	18	329
User 8	0.902564	0.318182	0.233333	0.269231	0.041667	0.766667	7	15	23	345
User 9	0.928205	0.55	0.366667	0.44	0.025	0.633333	11	9	19	351
User 10	0.902564	0.333333	0.266667	0.296296	0.044444	0.733333	8	16	22	344
User 12	0.928205	0.53125	0.566667	0.548387	0.041667	0.433333	17	15	13	345
User 13	0.987179	0.903226	0.933333	0.918033	0.008333	0.066667	28	3	2	357
Avg.	91%	41%	42%	41%	5%	58%				

Accelerometer Feature Set

User	accuracy	Precision	Recall	F-Score	FP Rate	FN Rate	TP	FP	FN	TN
User 3	0.989744	0.964286	0.9	0.931034	0.002778	0.1	27	1	3	359
User2	0.966667	0.757576	0.833333	0.793651	0.022222	0.166667	25	8	5	352
User1	0.920513	0.48	0.4	0.436364	0.036111	0.6	12	13	18	347
User 5	0.905128	0.387097	0.4	0.393443	0.052778	0.6	12	19	18	341
User 6	0.94359	0.642857	0.6	0.62069	0.027778	0.4	18	10	12	350
User 7	0.930769	0.545455	0.6	0.571429	0.041667	0.4	18	15	12	345
User 11	0.941026	0.62069	0.6	0.610169	0.030556	0.4	18	11	12	349
User4	0.966667	0.84	0.7	0.763636	0.011111	0.3	21	4	9	356
User 8	0.953846	0.6875	0.733333	0.709677	0.027778	0.266667	22	10	8	350
User 9	0.930769	0.571429	0.4	0.470588	0.025	0.6	12	9	18	351
User 10	0.917949	0.478261	0.733333	0.578947	0.066667	0.266667	22	24	8	336
User 12	1	1	1	1	0	0	30	0	0	360
User 13	0.930769	0.551724	0.533333	0.542373	0.036111	0.466667	16	13	14	347
Avg.	95%	66%	65%	65%	3%	35%				

Appendix I: Random Forest Raw Data for Relevant Features

Full Feature Set

User	accuracy	Precision	Recall	F-Score	FP Rate	FN Rate	TP	FP	FN	TN
User 3	0.994872	0.966667	0.966667	0.966667	0.002778	0.033333	29	1	1	359
User2	0.984615	0.875	0.933333	0.903226	0.011111	0.066667	28	4	2	356
User1	0.958974	0.75	0.7	0.724138	0.019444	0.3	21	7	9	353
User 5	0.984615	0.9	0.9	0.9	0.008333	0.1	27	3	3	357
User 6	0.979487	0.84375	0.9	0.870968	0.013889	0.1	27	5	3	355
User 7	0.948718	0.625	0.833333	0.714286	0.041667	0.166667	25	15	5	345
User 11	0.971795	0.851852	0.766667	0.807018	0.011111	0.233333	23	4	7	356
User4	0.984615	0.928571	0.866667	0.896552	0.005556	0.133333	26	2	4	358
User 8	0.964103	0.807692	0.7	0.75	0.013889	0.3	21	5	9	355
User 9	0.974359	0.875	0.75	0.807692	0.008287	0.25	21	3	7	359
User 10	0.95641	0.69697	0.766667	0.730159	0.027778	0.233333	23	10	7	350
User 12	0.994872	0.966667	0.966667	0.966667	0.002778	0.033333	29	1	1	359
User 13	1	1	1	1	0	0	30	0	0	360
Avg.	98%	85%	85%	85%	1%	15%				

Blink Feature Set

User	accuracy	Precision	Recall	F-Score	FP Rate	FN Rate	TP	FP	FN	TN
User 3	0.992308	0.9375	0.967742	0.952381	0.005571	0.032258	30	2	1	357
User2	0.979487	0.76	0.904762	0.826087	0.01626	0.095238	19	6	2	363
User1	0.941026	0.658537	0.75	0.701299	0.039548	0.25	27	14	9	340
User 5	0.95641	0.621622	0.884615	0.730159	0.038462	0.115385	23	14	3	350
User 6	0.984615	0.892857	0.892857	0.892857	0.008287	0.107143	25	3	3	359
User 7	0.958974	0.685714	0.827586	0.75	0.030471	0.172414	24	11	5	350
User 11	0.961538	0.757576	0.78125	0.769231	0.022346	0.21875	25	8	7	350
User4	0.974359	0.714286	0.789474	0.75	0.016173	0.210526	15	6	4	365
User 8	0.95641	0.6	0.571429	0.585366	0.02168	0.428571	12	8	9	361
User 9	0.95641	0.615385	0.695652	0.653061	0.027248	0.304348	16	10	7	357
User 10	0.958974	0.666667	0.72	0.692308	0.024658	0.28	18	9	7	356
User 12	0.979487	0.78125	0.961538	0.862069	0.019231	0.038462	25	7	1	357
User 13	0.984615	0.818182	1	0.9	0.016529	0	27	6	0	357
Avg.	97%	73%	83%	77%	2%	17%				

Induced Blink Feature Set

User	accuracy	Precision	Recall	F-Score	FP Rate	FN Rate	TP	FP	FN	TN
User 3	0.997436	1	0.966667	0.983051	0	0.033333	29	0	1	360
User2	0.969231	0.6875	0.916667	0.785714	0.027322	0.083333	22	10	2	356
User1	0.935897	0.609756	0.735294	0.666667	0.044944	0.264706	25	16	9	340
User 5	0.971795	0.733333	0.88	0.8	0.021918	0.12	22	8	3	357
User 6	0.964103	0.65625	0.875	0.75	0.030055	0.125	21	11	3	355
User 7	0.961538	0.69697	0.821429	0.754098	0.027624	0.178571	23	10	5	352
User 11	0.951282	0.647059	0.758621	0.698413	0.033241	0.241379	22	12	7	349
User4	0.974359	0.714286	0.789474	0.75	0.016173	0.210526	15	6	4	365
User 8	0.94359	0.458333	0.55	0.5	0.035135	0.45	11	13	9	357
User 9	0.953846	0.541667	0.65	0.590909	0.02973	0.35	13	11	7	359
User 10	0.958974	0.64	0.695652	0.666667	0.024523	0.304348	16	9	7	358
User 12	0.971795	0.69697	0.958333	0.807018	0.027322	0.041667	23	10	1	356
User 13	0.984615	0.8125	1	0.896552	0.016484	0	26	6	0	358
Avg.	96%	68%	82%	74%	3%	18%				

Standardized Blink Feature Set

User	accuracy	Precision	Recall	F-Score	FP Rate	FN Rate	TP	FP	FN	TN
User 3	0.997436	1	0.961538	0.980392	0	0.038462	25	0	1	364
User2	0.969231	0.655172	0.904762	0.76	0.0271	0.095238	19	10	2	359
User1	0.935897	0.6	0.727273	0.657534	0.044818	0.272727	24	16	9	341
User 5	0.971795	0.703704	0.863636	0.77551	0.021739	0.136364	19	8	3	360
User 6	0.964103	0.685714	0.888889	0.774194	0.030303	0.111111	24	11	3	352
User 7	0.961538	0.655172	0.791667	0.716981	0.027322	0.208333	19	10	5	356
User 11	0.951282	0.636364	0.75	0.688525	0.033149	0.25	21	12	7	350
User4	0.974359	0.73913	0.809524	0.772727	0.01626	0.190476	17	6	4	363
User 8	0.94359	0.48	0.571429	0.521739	0.03523	0.428571	12	13	9	356
User 9	0.953846	0.56	0.666667	0.608696	0.02981	0.333333	14	11	7	358
User 10	0.958974	0.64	0.695652	0.666667	0.024523	0.304348	16	9	7	358
User 12	0.971795	0.72973	0.964286	0.830769	0.027624	0.035714	27	10	1	352
User 13	0.984615	0.793103	1	0.884615	0.016349	0	23	6	0	361
Avg.	96%	68%	82%	74%	3%	18%				

Movement Feature Set

User	accuracy	Precision	Recall	F-Score	FP Rate	FN Rate	TP	FP	FN	TN
User 3	0.994872	0.965517	0.965517	0.965517	0.00277	0.034483	28	1	1	360
User2	0.989744	0.923077	0.923077	0.923077	0.005495	0.076923	24	2	2	362
User1	0.941026	0.5625	0.666667	0.610169	0.038567	0.333333	18	14	9	349
User 5	0.961538	0.571429	0.842105	0.680851	0.032345	0.157895	16	12	3	359
User 6	0.969231	0.7	0.875	0.777778	0.02459	0.125	21	9	3	357
User 7	0.930769	0.47619	0.8	0.597015	0.060274	0.2	20	22	5	343
User 11	0.961538	0.703704	0.730769	0.716981	0.021978	0.269231	19	8	7	356
User4	0.979487	0.857143	0.857143	0.857143	0.01105	0.142857	24	4	4	358
User 8	0.938462	0.615385	0.727273	0.666667	0.042017	0.272727	24	15	9	342
User 9	0.976923	0.9	0.72	0.8	0.005479	0.28	18	2	7	363
User 10	0.958974	0.666667	0.72	0.692308	0.024658	0.28	18	9	7	356
User 12	0.987179	0.878788	0.966667	0.920635	0.011111	0.033333	29	4	1	356
User 13	0.997436	0.965517	1	0.982456	0.002762	0	28	1	0	361
Avg.	97%	75%	83%	78%	2%	17%				

Gyroscope Feature Set

User	accuracy	Precision	Recall	F-Score	FP Rate	FN Rate	TP	FP	FN	TN
User 3	0.971795	0.666667	0.952381	0.784314	0.0271	0.047619	20	10	1	359
User2	0.964103	0.571429	0.888889	0.695652	0.032258	0.111111	16	12	2	360
User1	0.933333	0.46875	0.625	0.535714	0.046448	0.375	15	17	9	349
User 5	0.941026	0.354839	0.785714	0.488889	0.053191	0.214286	11	20	3	356
User 6	0.958974	0.48	0.8	0.6	0.034667	0.2	12	13	3	362
User 7	0.948718	0.571429	0.8	0.666667	0.041096	0.2	20	15	5	350
User 11	0.935897	0.454545	0.681818	0.545455	0.048913	0.318182	15	18	7	350
User4	0.935897	0.34375	0.733333	0.468085	0.056	0.266667	11	21	4	354
User 8	0.930769	0.419355	0.590909	0.490566	0.048913	0.409091	13	18	9	350
User 9	0.953846	0.47619	0.588235	0.526316	0.029491	0.411765	10	11	7	362
User 10	0.933333	0.40625	0.65	0.5	0.051351	0.35	13	19	7	351
User 12	0.971795	0.655172	0.95	0.77551	0.027027	0.05	19	10	1	360
User 13	0.997436	0.966667	1	0.983051	0.00277	0	29	1	0	360
Avg.	95%	53%	77%	62%	4%	23%				

Accelerometer Feature Set

User	accuracy	Precision	Recall	F-Score	FP Rate	FN Rate	TP	FP	FN	TN
User 3	0.997436	1	0.964286	0.981818	0	0.035714	27	0	1	362
User2	0.969231	0.74359	0.935484	0.828571	0.027855	0.064516	29	10	2	349
User1	0.958974	0.72	0.666667	0.692308	0.019284	0.333333	18	7	9	356
User 5	0.969231	0.7	0.875	0.777778	0.02459	0.125	21	9	3	357
User 6	0.961538	0.454545	0.769231	0.571429	0.03183	0.230769	10	12	3	365
User 7	0.941026	0.485714	0.772727	0.596491	0.048913	0.227273	17	18	5	350
User 11	0.958974	0.64	0.695652	0.666667	0.024523	0.304348	16	9	7	358
User4	0.976923	0.833333	0.862069	0.847458	0.01385	0.137931	25	5	4	356
User 8	0.94359	0.59375	0.678571	0.633333	0.035912	0.321429	19	13	9	349
User 9	0.966667	0.76	0.730769	0.745098	0.016484	0.269231	19	6	7	358
User 10	0.946154	0.548387	0.708333	0.618182	0.038251	0.291667	17	14	7	352
User 12	0.992308	0.933333	0.965517	0.949153	0.00554	0.034483	28	2	1	359
User 13	0.979487	0.692308	1	0.818182	0.021505	0	18	8	0	364
Avg.	97%	70%	82%	75%	2%	18%				

Appendix J: Flashing the Glass

Firmware Troubleshooting

Reason for Flashing:

When we first received our Google Glass, if you turned it on by pressing the power button for a moment, it would open a starting screen "GLASS", pause on that screen for a moment, and then restart and show the glass logo again. Additionally, the device would heat up while continuing this loop. We suspected the firmware had been updated to an incompatible version at some point, so we decided flashing the device with a new version of the Glass kernel would resolve the issue. Flashing, as a process on Google Glass, clears out the old firmware on the Glass device and replaces it with working firmware. Additionally, it deletes all applications installed on the device as well as any stored memory.

Failed Flashing Attempts:

During our multiple attempts to flash our Google Glass device, Macintosh OS X was found to be unsupported for the required Fastboot commands. In particular, we observed that when we plugged the Glass into a computer running Macintosh OS X, the Macintosh computer recognized the Glass device when we called fastboot devices but hung when we called fastboot oem unlock. Upon further investigation, we found that at the time of our attempt, this was an existing, unsolved bug that had existed for Macintosh OS X ever since XE11.

Linux Ubuntu: Similar to the process performed on the Macintosh computer, we put Glass in Fastboot mode and connected it to a computer running Ubuntu Linux. Again, the Glass device was listed under *fastboot devices*. We then proceeded to run the *fastboot oem unlock* command. That command was then followed by the wipe command. However, the flashing process hung on the wipe command. The wipe ran for a few hours. Next, we ran the flash command which ran overnight and ended. The Glass device was still not working at this point. No results were ultimately found as to why this happened.

Linux Lint: This operating system hung on the *flash oem unlock* command.

Falshing was successful on Windows 7.

ADB and Fastboot are Android modification programs. You can use them for Android phones as well as for Google Glass (http://dottech.org/21534/how-to-install-adb-and-fastboot-on-your-windows-computer-for-use-with-your-android-phone/). We downloaded the ADB and Fastboot for Windows from

this cited link and installed them into our computers environment following the directions described from the same link.

Flashing the Glass

The Google Glass Developer site describes the process for flashing to a new factory image (https://developers.google.com/glass/tools-downloads/system). We decided to download the version 19.1 image pack because our eclipse setup used API version 19.

Before we could run the Fastboot commands on the device, we needed to edit the device driver in the glass package to recognize our input device (http://stackoverflow.com/questions/16928983/google-glass-adb-devices-doesnt-find-omap4430-driver-not-installed-cant-find). Following these directions we updated our android_winusb.info file found in the android sdk folder(…\sdk\extras\google\usb_driver). We changed four lines in this file. Two under the Google.NTx86 and the other two under the Google.NTamd64 section. We changed the lines in this section to match the driver identification of our current input device.

The Google Developer's Guide suggests we run these commands in order:

$ adb reboot bootloader # enter fastboot mode

$ fastboot devices # verify device is in fastboot, should see its serial no.

$ fastboot flash boot boot.img # flash partitions with factory images

$ fastboot flash system system.img

$ fastboot flash recovery recovery.img

$ fastboot erase cache # optional, erase the cache and userdata partition

$ fastboot erase userdata

$ fastboot oem lock # optional, only if you want to re-lock the bootloader.

We ran these commands from the folder where we downloaded the kernel image. See "Results from Flashing Commands" in this section.

Post Flash:

After we ran all of these commands and received termination status from the lock command, we unplugged the Glass from our computer. The Glass then restarted twice on its own. It hung on the logo screen for a few minutes and then entered the default Glass setup video for version 19. We plugged the glass into the computer again after setting the Glass up, and it installed the newest version of the Glass kernel (version 21.3) successfully.

Results from Flashing Commands

Ran "fastboot flash boot boot.img" from the GlassImgs folder

Received:

sending 'boot' (5386 KB)...
OKAY [0.238s]
writing 'boot'...
OKAY [0.980s]
finished. total time: 1.226s

Ran "fastboot flash system system.img" from the GlassImgs folder

Received:

sending 'system' (636374 KB)...
OKAY [26.795s]
writing 'system'...
OKAY [83.940s]
finished. total time: 110.742s

Ran "fastboot flash recovery recovery.img" from the GlassImgs folder

Received:

sending 'recovery' (6400 KB)...
OKAY [0.284s]
writing 'recovery'...
OKAY [1.973s]
finished. total time: 2.265s

Ran "fastboot erase cache"

Received:

erasing 'cache'...
OKAY [0.251s]
finished. total time: 0.255s

Ran "fastboot erase userdata"

Received:

erasing 'userdata'...
OKAY [0.252s]
finished. total time: 0.256s

Ran "fastboot oem lock"
Received:

...
(bootloader) Device locked!
OKAY [0.005s]
finished. total time: 0.007s

Appendix K: Settings for the Glass

Table 10: Experimental setup for glass settings

Feature	Setting
Keep Screen On While Charging	Enabled
Show Layout Bounds & Margins	Disabled
Show GPU Overdraw	Disabled
Animation Time Scale Factor	1X
Layout Screen Overlay	Disabled
Auto Backup	Enabled
Head Wakeup	Enabled: 30 Degrees
Head Nudge	Enabled
On head Detection	Enabled
Screen Lock	Enabled
Notification Glance	Enabled
Wink for Picture	Disabled
Volume	50%

Appendix L: The Android Debugging Bridge (ADB)

The Android Debug Bridge is a built-in tool designed to facilitate the running and testing of applications on Android devices. In addition to loading applications and streaming log data over a USB connection, ADB has several other useful features that we used for this project. The first is using ADB to gain root access to the Google Glass. This allows us to access parts of the system normally off limits to users. ADB also allows us to access device using a bash-like shell. We used this when setting up our low-level blink capture system. Finally, ADB allows us to forward UDB and TDP ports over a USB connection. This allows us to test our system locally without needing a network connection.

Appendix M: Rooting Google Glass

We rooted the Glass in order to facilitate blink detection. The blink detection built in to Google Glass is designed to capture deliberate, slower blinks, rather than the faster natural blinks we are trying to detect. A rooted Glass gives us access to the low level sensors needed to implement the blink capture system described in the “Blink of an Eye” paper in the related work section.

Rooting the glass is very similar to flashing the glass. In order to root the glass, begin by running the command “adb root”. This will restart adb with as root user for the Google Glass. Next, follow the instructions for flashing the Google Glass.