Continuous Authentication with Touch Behavioral Biometrics and Voice on Wearable Glasses

Ge Peng, Gang Zhou, *Senior Member, IEEE,* David T. Nguyen, Xin Qi,
Qing Yang, and Shuangquan Wang, *Member, IEEE*

***Abstract*—Wearable glasses are on the rising edge of development with great user popularity. However, user data stored on these devices brings privacy risks to the owner. To better protect the owner's privacy, a continuous authentication system is needed. In this paper, we propose a continuous and noninvasive authentication system for wearable glasses, named GlassGuard. GlassGuard discriminates the owner and an impostor with behavioral biometrics from 6 types of touch gestures (single-tap, swipe forward, swipe backward, swipe down, two-finger swipe forward, and two-finger swipe backward) and voice commands, which are all available during normal user interactions. With data collected from 32 users on Google Glass, we show that GlassGuard achieves 99% detection rate and 0.5% false alarm rate after 3.5 user events on average when all types of user events are available with equal probability. Under five typical usage scenarios, the system has a detection rate above 93% and a false alarm rate below 3% after less than 5 user events.**

***Index Terms*—continuous authentication; noninvasive; behavioral biometric; voice authentication; wearable glasses;**

I. Introduction

Wearable glasses have attracted considerable attention over the years. More and more large companies are investing money on wearable glasses. Now, more than 20 wearable glasses are under production or development [1], including Google Glass, Microsoft HoloLens, Facebook Oculus Rift, Epson Moverio, Sony SmartEyeglass, Intel Radar Pace, and Osterhout Design Group (ODG) R-7. The hand-free nature and augmented reality capability of wearable glasses open up new opportunities for human-machine interactions. Wearable glasses are going to become an important part of our daily lives. A recent study by Juniper Research shows that more than 12 million consumer smart glasses will be shipped in 2020, increasing from less than one million in 2016 [2].

When using these wearable glasses, some personal information is stored on the devices for easy revisit, such as contacts information, location data, messages, emails, personal photos and videos, account information, and much more. When the owner takes off his/her smart glasses and puts them aside, for example when the device is charging or when the owner needs to go to the restroom, an impostor will certainly have the chance to grab the device and access the owner's private information. In addition to privacy leakage of the owner, an imposter can also send e-mails/messages to any contact stored on the glasses in the guise of the owner, bringing privacy threats to the owner's friends, family, and colleagues.

To protect user privacy on wearable glasses, a continuous authentication system is more suitable than a one-time authentication system. A one-time user authentication system only authenticates a user when he/she tries to unlock the device, typically by asking the user to input a password or PIN, a graphical pattern, or a sequence of touch gestures (a user is authorized as long as the right gesture types are performed in the correct order). However, the owner may forget to lock the device right after using the device. There are mechanisms which automatically lock the device upon an event, such as screen timeout. Google Glass has on-head detection, which automatically locks the device when a user takes off the glass. However, on-head detection on Google Glass is not reliable. It does not work when the glass is not worn in the perfect position. It also does not work with Google Glass frames which are customized for users in need of vision correction. More importantly, even if the device is locked, a one-time authentication system can easily be broken into by peeking [3], [4], [5] or smudge attacks [6], [7]. Alternatively, wearable glasses can automatically pair with another trusted device, such as the owner's smartphone, to perform authentication. However, successful pairing only indicates that the owner is nearby. It does not necessarily mean that the current user is the owner. A one-time authentication solution does not work well. Therefore, a continuous authentication system which continuously authenticates the user during the whole time of user operation is needed to better protect user privacy.

Touch behavioral biometrics have been demonstrated to be effective in continuous user authentication on smartphones [8], [9], [10]. The hypothesis is that different users have different characteristics when interacting with smartphones and these behavioral biometrics are difficult to fake. We believe the same is also true on wearable glasses. However, due to user interaction differences between wearable glasses and smartphones, systems proposed on smartphones cannot be applied directly on wearable glasses. First, users hold their smartphones with hand(s) but wear smart glasses on their head. Motion sensors, such as accelerometers and gyroscopes, respond to user touch events in a different pattern on wearable glasses. As a result, features working on smartphones may not work well on wearable glasses. Second, wearable glasses only have a touchpad with no virtual keyboard support. Keystroke biometric

This work was supported in part by the U.S. National Science Foundation under grant CNS-1250180 and CNS-1253506 (CAREER).

Ge Peng, Gang Zhou, Qing Yang, and Shuangquan Wang are with College of William and Mary, Williamsburg, VA 23185 USA (e-mail: {gpeng, gzhou, qyang, swang}@cs.wm.edu.)

Xin Qi and David T. Nguyen were with College of William and Mary, Williamsburg, VA 23188 USA. Xin Qi is now with VMware Inc, Palo Alto, CA 94304 USA (e-mail:xqi@email.wm.edu). David T. Nguyen is now with Facebook Inc, Menlo Park, CA 94025 (email: dnguyen@cs.wm.edu)

information [11] is not available on wearable glasses. Third, wearable glasses touchpad is much smaller than smartphone touch screen. Thus, the resolution of biometric information on wearable glasses, such as coordinates, is much lower than that on smartphones. Feature discriminability needs to be examined on wearable glasses. Finally, different touch gestures are used on wearable glasses. For example, there are no pinch gestures on Google Glass. To zoom in or out, a two-finger swipe forward or backward gesture is used. Thus, new features that are specific to wearable glasses need to be explored.

In this paper, we study the performance of using touch gestures and voice commands for continuous user authentication on wearable glasses with the example of Google Glass. We consider both touch gestures and voice commands as they are two major channels for user interaction on wearable glasses. An authentication system based only on touch biometrics can be easily circumvented by using voice commands. Similarly, a system purely based on voice authentication does not always work, as voice commands are not available all the time. A user may be in a situation when speaking is not appropriate, e.g., at a meeting with quiet surroundings. Although touch behavioral based authentication [8], [9], [10] and voice based authentication [12], [13], [14] are two well-studied fields, our contributions lie in that we study them in a new platform and we integrate these two dimensions to accommodate various scenarios.

In our system, we use both touch behavioral features extracted from touchpad data as well as corresponding sensor data during touch gestures and voice features extracted from user-issued voice commands. These features can be easily extracted in the background when users normally interact with wearable glasses. It does not require extra efforts from users. Thus, our system works in a noninvasive way. Note that in this paper, we focus on one specific model of wearable glasses: Google Glass. However, the method introduced and the authentication system framework proposed in this paper also apply to other wearable glasses with a touch panel and built-in microphone and speakers, such as SiME Smart Glasses [15], Recon Jet [16], and Vuzix M300 [17].

We summarize our contributions as follows.

- We conduct a user study on Google Glass and collect user interaction data from 32 human subjects. The data we collect includes touch event data with corresponding sensor readings, and voice commands. Six types of gestures are covered in the study: single-tap, swipe forward, swipe backward, swipe down, two-finger swipe forward, and two-finger swipe backward.
- With the data collected, we define and extract 99 behavioral features for one-finger touch gestures, 156 features for two-finger touch gestures, and 19 voice features for user voice commands. We evaluate the discriminability of these features with one-class Support Vector Machine (SVM) model for user authentication purpose on Google Glass.
- We design a simple but effective online user authentication system for wearable glasses, namely GlassGuard, which works in a continuous and noninvasive manner. GlassGuard employs a mechanism adapted from Threshold Random Walking (TRW) to make a decision from multiple user events only when it is confident. Our preliminary results indicate that it achieves high accuracy with acceptable delay.

The rest of this paper is organized as follows. First, we introduce the proposed features and evaluate the features with real user data in Section II. Then in Section III, we present our continuous authentication system, GlassGuard. We evaluate the system performance in Section IV and discuss related work in Section V. Finally, we draw our conclusions and discuss future work in Section VI.

II. Features for Continuous User Authentication

In this section, we first introduce all the features that we are going to study. Then, we describe a user study that we have carried out to collect real user interaction data. With the data collected, we evaluate the performance of these features and conduct feature selection.

A. Key User Events

Common touch gestures on Google Glass are as follows: single-tap to select an item, swipe backward (forward) to move left (right) through items, swipe down to go back, and two-finger swipe forward (backward) to zoom in (out).

With a built-in microphone and speaker, Google Glass accepts voice commands as user inputs, such as "OK, Glass! Take a picture!" This offers a hand-free interaction which can be extremely useful for people with disabilities and for wearers with both hands busy.

When designing features, we focus on the above six types of touch gestures and all voice commands.

B. Proposed Features

We propose different feature sets for one-finger touch gestures, two-finger touch gestures, and voice commands. Our features for one-finger touch gestures are proposed based on several existing works on smartphones[8], [18], [19], as here we only want to obtain a list of potential features. Later, we conduct feature selection to find the best features that work on Google Glass.

Features for One-finger Touch Gestures. We divided our features for touch gestures into two categories: (1) touch-based features, which are features extracted from touchpad data; and (2) sensor-based features, which are features extracted from sensor readings during touch gestures. Figure 1 gives an example of a one-finger touch gesture along the timeline. Table I lists all 18 touch-based features for a one-finger touch gesture. Note that each touch gesture generates multiple records in the raw touch data, as the touchpad is continuously sampling. The statistics below, such as minimum, maximum, and median, are calculated from multiple samples of one touch gesture starting at time t_{start} and ending at time t_{end}.

Let x, y, z be sensor readings (accelerometer, or gyroscope, or magnetometer) in each axis and $net = \sqrt{x^2 + y^2 + z^2}$. Table II lists all sensor-based features during a one-finger touch gesture.

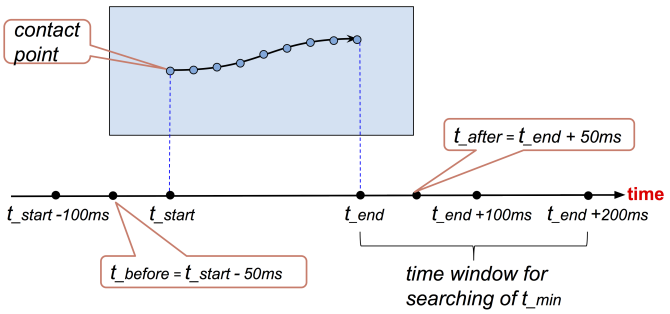


Fig. 1: An example of a one-finger touch gesture

TABLE I: Touch-based features

Aspect	Feature	Explanation
time	*duration*	time difference between the first and last records of a touch gesture
distance	*distance* *distance_x* *distance_y*	distance between contact points of the first and last records of a touch gesture, and its values along *x*-axis and *y*-axis
speed	*speed* *speed_x* *speed_y*	speed of finger movement on touchpad during a touch gesture, and its value along *x*-axis and *y*-axis
pressure	{*mean, max, min, median, stdev*}*_pressure*	mean, max, min, median, and standard deviation of pressure values during a touch gesture
	{*q1, q2, q3*}*_pressure*	the 25%, 50%, and 75% quartiles of pressure values during a touch event
	first_pressure *last_pressure*	pressure value of the first and last records respectively
	max_pressure_por	time portion to achieve the maximum value of pressure. $= (t_m - t_{start})/(t_{end} - t_{start})$ where t_m is the timestamp for the record with the maximum pressure value.

Let ϕ_{max} be the maximum accelerometer readings during a touch event and t_{max} be the corresponding timestamp. Considering a 100ms time window before a touch gesture, let t_{before} be the center of the time window and ϕ_{before} be the average net value of accelerometer readings. Considering a 100ms time window after a touch gesture, let t_{after} be the center of the time window and ϕ_{after} be the average net value of accelerometer readings. Then, in Table II, we list the sensor-based features. The following features in Table II are calculated as

$$acc\_after\_before\_net = \phi_{after} - \phi_{before} \quad (1)$$

$$acc\_mean\_before\_net = acc\_mean\_net - \phi_{before} \quad (2)$$

$$acc\_max\_before\_net = acc\_max\_net - \phi_{before} \quad (3)$$

$$acc\_ndt\_before\_after = \frac{t_{after} - t_{before}}{\phi_{after} - \phi_{before}} \quad (4)$$

$$acc\_ndt\_max\_after = \frac{t_{after} - t_{max}}{\phi_{after} - \phi_{max}} \quad (5)$$

$$acc\_time\_to\_restore = t_{min} - t_{end} \quad (6)$$

where t_{min} is the time instance within a $T_2 = 200$ms time window after a touch event when the sensor reading restores to the average value before this touch event.

$$t_{min} = \underset{t_j \in (t_{end}, t_{end}+T_2]}{\arg\min} \left| net_{t_j} - \phi_{before} \right| \quad (7)$$

TABLE II: Sensor-based features

Aspect	Feature	Explanation
absolute value	*acc_mean_*{*x, y, z, net*}	**mean** values of sensor readings during a touch gesture
	*acc_median_*{*x, y, z, net*}	**median** values of sensor readings during a touch gesture
	*acc_std_*{*x, y, z, net*}	**standard deviations** of sensor readings during a touch gesture
change in value	*acc_after_before_*{*x, y, z, net*}	change of **average** sensor readings **after** a touch gesture compared to that **before** the touch gesture. See Eq. (1)
	*acc_mean_before_*{*x, y, z, net*}	the difference between the **average** sensor readings **during** a touch gesture and that **before** the touch gesture. See Eq. (2)
	*acc_max_before_*{*x, y, z, net*}	the difference between the **max** sensor readings **during** a touch gesture and the **average** sensor readings **before** the touch gesture. See Eq. (3)
time to change	*acc_ndt_before_after*	the normalized time duration for the **average** sensor readings to change from a state before a touch event ϕ_{before} to a state after the touch event ϕ_{after}. See Eq. (4)
	acc_ndt_max_after	the normalized time duration for sensor readings to change from the **max** value ϕ_{max} during a touch event to a state after the touch event ϕ_{after}. See Eq.(5)
	acc_time_to_restore	time duration after a touch event for sensor readings to **restore** to average value before the touch event. See Eq. (6)

In order to save space, we only list the 27 features based on accelerometer data. Features based on gyroscope and magnetometer are defined accordingly. In total, we have 81 sensor-based features. We do not include frequency domain features here as extracting frequency domain features is energy hungry and requires high computation capability. Later in Section IV, we show that our system achieves high accuracy with only these time domain features.

Features for Two-finger Touch Gestures. Two-finger touch gestures have two contact points on the touchpad. For each contact point, we define a set of touch-based features presented above for one-finger touch gestures. For example, duration for each contact point (denoted as $duration_1$, $duration_2$) and distance for each contact point (denoted as $distance_1$, $distance_2$). Moreover, the relative information between the two contact points may also be useful for user authentication. For two-finger touch gestures, we design the following 29 touch-based features additionally.

- *duration*: the duration of a touch gesture. It may be different from both $duration_1$ and $duration_2$ when the first and last records of a touch gesture belong to different fingers.

- mean, max, min, median, and standard deviation of distances (or distances along x axis, or distances along y axis) between two contact points.
- *q1_diff_dist*, *q2_diff_dist*, *q3_diff_dist*: The 25%, 50%, and 75% quartiles of distances between two contact points.
- *first_diff_dist*, *last_diff_dist*: the distance between the first (last) contact points of two fingers.
- The difference between the mean (or maximum, or minimum) pressure values for two fingers during a touch gesture.
- The maximum difference between pressure values of two fingers during a touch gesture.
- The minimum difference between pressure values of two fingers during a touch gesture.
- The difference between speeds (or speeds along the x axis, or speeds along the y axis) of two fingers.

Sensor-based features for two-finger touch gestures are the same as those for one-finger touch gestures. So, we have proposed 156 features in total for two-finger touch gestures, 81 from sensor data and 75 from touch data.

Features for Voice Commands. For voice features, we use Mel-Frequency Cepstral Coefficients (MFCC). MFCC is one of the most effective and widely used features in speech processing [20]. An audio file is recorded from each voice command. The audio file is then divided into frames with a sliding window of 25 ms and a step size of 10 ms. For each frame, we extract 20 coefficients. The first coefficient indicates the direct current of the voice signal. It does not convey any information about the spectral shape. So, we discard the first coefficient [14] and use the 2nd to the 20th coefficients to construct an MFCC vector. Before extracting MFCC vectors, we apply silence removal [21] to the audio.

C. User Study

In order to evaluate the features listed above, we conduct a user study to collect real user data[1].

To obtain all the touch event data, we use the "getevent" tool [22]. With this tool, we are able to collect raw touch data (including coordinates of contact points, and pressure) at background without user perception.

To obtain sensor readings during a touch event, we write a Google Glass application that samples sensor data with system API. The application runs as a background service. It does not interrupt users' normal operations. The application logs down data from all three inertial sensors (accelerometer, magnetometer, and gyroscope) with a sampling rate of 200 Hz, 200 Hz, and 100 Hz, respectively.

Google Glass automatically records user commands and saves them locally. To analyze these voice commands, we pull these files out from Google Glass with adb [23] tool.

Using the tools introduced above, we conducted a user study and collected interaction data from 32 subjects. All of the participants are college students, comprising 13 females and 19 males. The data of each user is collected from multiple sessions in a two-hour time frame. In order to collect as many interested user events as possible, a user is asked to perform a specific task in each session. There are 7 tasks in the user study and each task is repeated multiple times: (1) swipe to view the application list one by one; (2) swipe to view the options in the settings menu one by one; (3) take pictures with touch gestures; (4) take pictures with voice commands; (5) Google search with voice commands; (6) delete pictures one by one; (7) use a customized application which asks the user to performance a series of randomly selected touch gestures. We carried out our user study on a Google Glass with system version XE 18.11. All data is collected in the background while users are standing and interacting with Google Glass normally. Table III shows the amount of the data collected.

TABLE III: Amount of the data collected

Touch data				
	Mean	Max	Min	Sum
# of single-tap	466.3	576	344	14281
# of swipe forward	629.0	1031	484	20127
# of swipe backward	599.7	862	433	19190
# of swipe down	483.0	615	354	15457
# of two-finger swipe forward	549.3	919	353	17576
# of two-finger swipe backward	534.3	722	341	17098
Sensor data				
Accelerometer (in MB)	45	66	35	1454
Gyroscope (in MB)	50	73	38	1599
Magnetometer (in MB)	23	33	18	733
Audio (voice commands)				
	Mean	Max	Min	Sum
# of audio	39.06	52	17	1250
Length of audios (in seconds)	2.28	15	0.95	2855.6

D. Feature Selection

We have proposed a set of features for each touch gesture. However, not all of them perform well in user authentication. We conduct feature selection to remove poor features and select features with high discriminability. By doing this, we also reduce the number of features needed, cutting down the computation cost of the online authentication system. The algorithm we use is Sequential Forward Search [24], and the classification Equal Error Rate (EER) is used as the criterion function.

Users have different characteristics when interacting with Google Glass. Some features work for all users as everyone is different in those aspects. Some features only work for a specific user because the owner has his (her) own peculiarity discriminating himself (herself) from others. So, we conduct user-specific feature selection: repeat the feature selection process 32 times, each time for a different user. Figure 2 shows the performance ranking of these features when only 5 features are used in each model. The number on the left side of a bar indicates the rank of a feature. The number on the right side of a bar shows the number of models which have used this feature, followed by the name of the feature. We have the following observations from the figures. First, *max_pressure* performs well for all one-finger touch gestures. Second, the minimum distance between two fingers, *min_diff_dist*, is the best feature for two-finger touch gestures.

[1]This user study was approved by the Protection of Human Subject Committee at College of William and Mary.

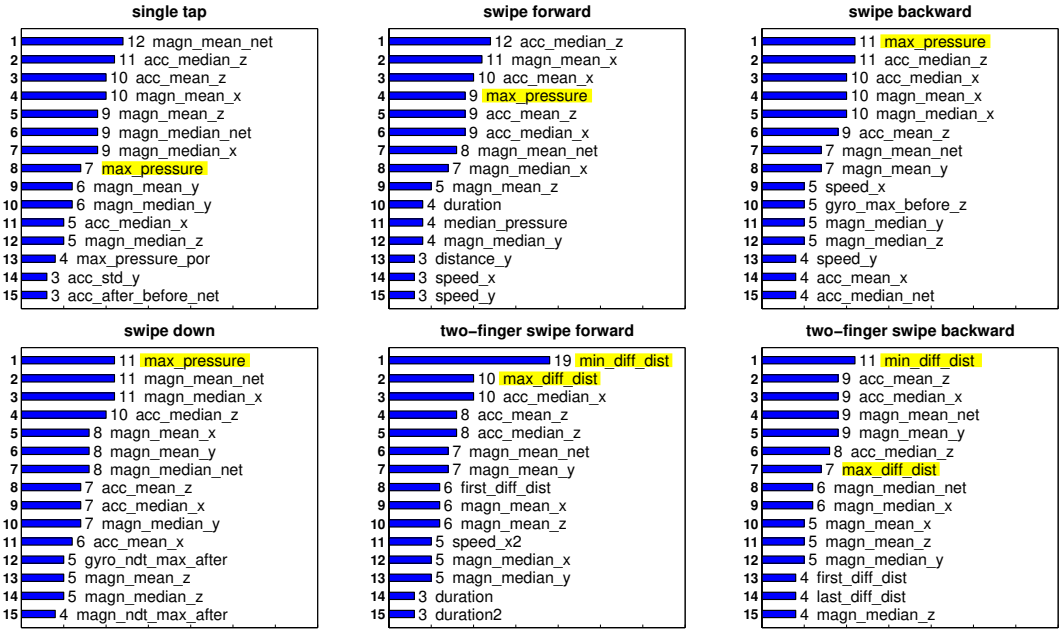


Fig. 2: The best 15 features overall when five best features are used in the model for each user. The number on the left side of a bar indicates the rank of the feature. The number on the right side of a bar shows the number of models which have used this feature, followed by the name of the feature.

The maximum distance between two fingers, *max_diff_dist*, is also among the tops. Third, accelerometer features and magnetometer features generally rank higher than gyroscope features. This also indicates that the device rotation is not as obvious as acceleration during touch events.

Comparing our features to those used on smartphones [8], [9], [10], [19], we have the following findings. (1) The touch size (area covered by fingertips) is an effective feature on smartphones. However, this information is not available on Google Glass. (2) The speed features of swipe gestures perform well on Google Glass, same as on smartphones. An exception on Google Glass is the swipe down gesture. This is because the vertical length of the touchpad here is much smaller than that on smartphones. (3) Two-finger swipe gestures are new gestures on Google Glass. Although there are also two-finger gestures (e.g. pinch) on smartphones, they have totally different definitions. Thus, different features are used. For example, the distance between two fingers are not useful for pinch gestures on smartphones. However, they perform pretty well for two-finger swipe gestures on Google Glass.

III. The GlassGuard System

In this section, we present the framework of our online authentication system, which we call GlassGuard. Figure 3 shows the architecture of the GlassGuard authentication system. There are five modules in the system. The Feature Extraction module calculates a set of features determined by offline training. In the following part of this section, we introduce each of the other four modules.

A. Event Monitor

The Event Monitor continuously monitors all user events when the screen is on, including touch events and voice

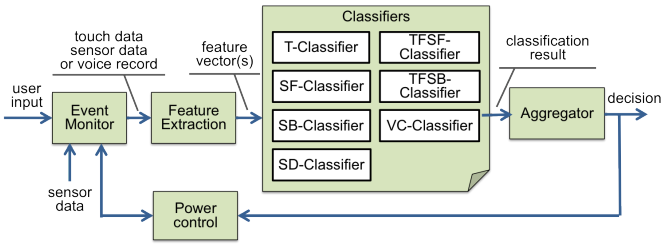


Fig. 3: System architecture of GlassGuard

commands. If it is a touch event, the Event Monitor forwards the touch data and the corresponding sensor data for feature extraction. If it is a voice command, the Event Monitor forwards the audio file for feature extraction.

The Event Monitor also communicates with the Power Control module. On one hand, it reports occurrences of user events to the Power Control module. On the other hand, it gets instructions from the Power Control module about whether it should forward data for feature extraction or not. The details are explained in the Power Control subsection.

B. Classifiers

After features are extracted, they are passed to one of the classifiers. To achieve high accuracy, we train one classifier for each gesture type and for voice command, respectively. There are seven classifiers in the system: T-Classifier for single-tap gestures, SF-Classifier for Swipe Forward gestures, SB-Classifier for Swipe Backward gestures, SD-Classifier for Swipe Down gestures, TFSF-Classifier for Two-Finger Swipe Forward gestures, TFSB-Classifier for Two-Finger Swipe Forward gestures, and VC-Classifier for Voice Commands.

To do user authentication, a classifier only needs to tell whether or not an observation belongs to the owner. In our system, an observation can be either a voice command or a touch event belonging to any of the aforementioned gesture types. In reality, a Google Glass only has observations from its owner, rather than impostors, for training. Thus, we use one-class SVM (Support Vector Machine) [25] as the model to do classification. We select SVM because it provides high accuracy and it is effective in high-dimensional spaces and flexible in modeling diverse sources of data [26], [27]. SVM has been demonstrated to perform well in detecting user patterns in various applications, such as mouse movement pattern [28], voice pattern [29], motion pattern [30], and user-generated network traffic pattern [31], and so on.

Touch Gesture Classifiers. We train the classifiers via ten-fold cross validation following the training routine suggested in the LIBSVM website [32]. To train a classifier for gesture type i, we divide all feature vectors of gesture type i into *positive* samples and *negative* samples. Positive samples are feature vectors from the user currently treated as the owner. Negative samples are feature vectors from all other users. We randomly divide all positive samples into k (k=10) equal size subsets, and do the same for negative samples. Then, we train a one-class SVM model with $k-1$ positive subsets, leaving one subset of positive samples for testing. Then, we test the same model with one subset of negative samples. We repeat the training and testing steps until each subset of positive samples and each subset of negative samples are used exactly once for testing. With all the decision values calculated from the SVM models, we plot the Receiver Operating Characteristic (ROC) curve, which is insensitive to class skew [33]. The mis-prediction ratio of all positive samples is False Reject Rate (FRR) and the mis-prediction ratio of all negative samples is False Accept Rate (FAR).

VC-Classifier. Classification of voice features (MFCC vectors) is done in the same way as classification for touch gestures. However, the FAR and FRR are calculated in a different way. To get the EER for the voice command classifier, we treat all MFCC vectors extracted from the same audio file as a whole. If the percentage of misclassified MFCC vectors in an audio file is greater than a threshold p, then we think this audio file is misclassified and treat this as one error. The FAR and FRR are calculated as percentage of misclassified audio files in owner's data and in other users' data, respectively. We do this because it is normal to treat one user voice command as one user event. A threshold p is used because the classification results of MFCC vectors are noisy as the audio contains background sound and notification sound of the glass system. The value of p can be experimentally decided.

C. Aggregator

GlassGuard has seven classifiers. All classifiers make predictions independently. For each user event, we obtain one classification result. Once a classification result is generated, it is passed to the Aggregator module. To improve the accuracy of the authentication system, the Aggregator combines multiple classification results, which may come from different classifiers, and makes one final decision: whether or not the current wearer is the owner. In order to do that, we need to solve two problems: (1) how to combine multiple classification results and (2) when to make decisions.

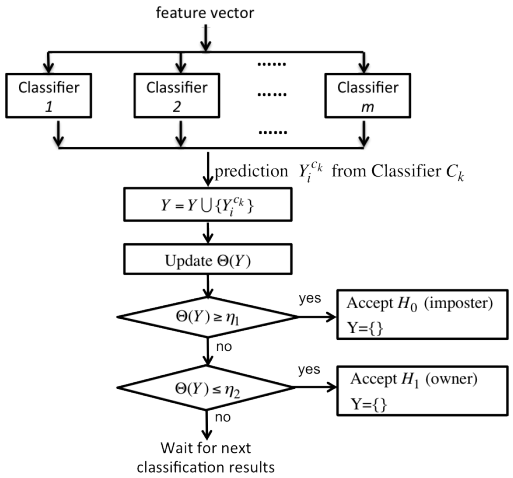


Fig. 4: Processing flow of the Aggregator module

In the GlassGuard system, the Aggregator employs a mechanism adapted from Threshold Random Walking (TRW) to make decisions when and only when it is confident. TRW is an online detection algorithm that has been successfully used to detect port scanning [34], botnets [35], and spam [36]. With TRW, predictions are made based on the likelihood ratio, which is the conditional probability of a series of classification results given the FAR and FRR of the classifier. When the likelihood ratio falls below a lower threshold, the system identifies the current user as an impostor. When the likelihood ratio reaches an upper threshold, the system identifies the current user as the owner. If the likelihood ratio is between the two thresholds, the system postpones making a prediction. In this paper, we choose TRW because it is simple but performs fast and accurately. However, TRW was originally designed to combine multiple results from a single classifier. In our system, we have multiple classifiers with different FARs and FRRs. We need to adapt TRW to accommodate multiple classifiers. Figure 4 shows the processing flow.

Assume that there are M classifiers ($M = 7$ in our GlassGuard system). For classifier c_k ($1 \leq k \leq M$), we have the estimated FAR_{c_k} and FRR_{c_k}. Suppose that at some point in time, we have gathered n classification results from the M classifiers, denoted as $Y = \{Y_i^{c_k} | 1 \leq i \leq n, 1 \leq k \leq M\}$, $Y_i^{c_k}$ is the i_{th} classification result and it is from classifier c_k. $Y_i^{c_k} = 1$ means classifier c_k predicts the event is from the owner. We call it a *positive* classification result. $Y_i^{c_k} = 0$ means classifier c_k predicts the event is not from the owner, which we call a *negative* classification result.

Let H_1 be the hypothesis that the current user is the owner and H_0 be the hypothesis that the current user is an impostor. Then the Aggregator calculates the following conditional probabilities

$$P(Y_i^{c_k} = 0|H_1) = FRR_{c_k}, \quad P(Y_i^{c_k} = 1|H_1) = 1 - FRR_{c_k}$$
$$P(Y_i^{c_k} = 1|H_0) = FAR_{c_k}, \quad P(Y_i^{c_k} = 0|H_0) = 1 - FAR_{c_k}$$

With n classification results and the above conditional probabilities for each classifier, the Aggregator calculates the likelihood ratio

$$\Theta(Y) = \prod_{i=1}^{n} \frac{P(Y_i^{c_k}|H_0)}{P(Y_i^{c_k}|H_1)} \tag{8}$$

In practice, both FAR and FRR are smaller than 50%. We have

$$\frac{P(Y_i^{c_k}=0|H_0)}{P(Y_i^{c_k}=0|H_1)} = \frac{1-FAR_{c_k}}{FRR_{c_k}} > 1$$

Similarly,

$$\frac{P(Y_i^{c_k}=1|H_0)}{P(Y_i^{c_k}=1|H_1)} = \frac{FAR_{c_k}}{1-FRR_{c_k}} < 1$$

which means, a negative classification result increases the value of $\Theta(Y)$ while a positive classification result decreases the value of $\Theta(Y)$.

When $\Theta(Y) \geq \eta_1$, the system takes hypothesis H_0 (is an impostor) to be true. When $\Theta(Y) \leq \eta_2$, the system takes hypothesis H_1 (is the owner) to be true. A basic principle of choosing values for η_1 and η_2 is [34]

$$\eta_1 = \frac{\beta}{\alpha} \qquad \eta_2 = \frac{1-\beta}{1-\alpha} \tag{9}$$

where α and β are two user-selected values. α is the expected false alarm rate (H_0 selected when H_1 is true) and β is the expected detection rate (H_0 selected when H_0 is true) of the whole system. The typical values are $\alpha = 1\%$ and $\beta = 99\%$.

D. Power Control

As with smartphones, energy consumption is an important user concern on Google Glass. In addition, if the power consumption of the system is too high, the temperature on the surface of Google Glass can easily get very high [37]. This may make users uncomfortable as well as slow down the system. So, while we aim to achieve high accuracy for the protection of the device owner's privacy, we also want to reduce the power consumption. In the GlassGuard system, the Power Control module is designed to improve the energy efficiency of the whole system. The basic idea is to pause feature extraction and classification whenever the privacy risk becomes low and restart those processes whenever the privacy risk reverts back to high.

When to pause? In the GlassGuard system, the Power Control module gets all decisions made by the Aggregator module and communicates with the Event Monitor module. If a negative decision (the current user is an impostor) is made by the Aggregator, then the glass system needs to do something to restrict access to the device, for example, lock the device and send an alert to the owner. The specific strategy to take when an impostor is detected is beyond the scope of this paper. Whenever a positive decision (the current user is the owner) is made by the Aggregator, the Power Control module instructs the Event Monitor module to temporarily pause forwarding data for feature extraction. As a result, data will not be processed by the Feature Extraction module or the Classifiers. To save more energy, when feature extraction is paused, the Event Monitor module also stops sampling sensor data.

When to restart? After sending a pause instruction to the Event Monitor module, the Power Control module starts a timer T for checking restarting conditions. T is set to a short interval, for example, 15 seconds. The Event Monitor module monitors all user events all the time and keeps updating the Power Control module with user activities. If there is no report of user events from the Event Monitor before timer T expires, it is possible that the user has been changed since the previous authentication decision. The Power Control module restarts feature extraction by instructing the Event Monitor module to continue forwarding data. As a result, feature extraction is enabled. The system extracts features and does all the following processing beginning from the next user event. If the Power Control module receives a report of user events from the Event Monitor module before timer T expires, it considers that the current user has not been changed since the previous positive decision from the Aggregator. The Power Control module does not restart feature extraction. At the same time, the timer T is reset. The basic assumption for this is that it is unlikely for the user to be changed in 15 seconds. And even if this happens, the owner should be able to instantly notice this as the owner was using the glass 15 seconds ago. In real world, there are cases when the owner wants to share something on the glass screen with his/her friends. The owner takes off the glass and passes it to his/her friends. In this case, the user is changed within a short time, perhaps less than 15 seconds. However, the owner knows this and the owner actually wants it to happen. Hence it does not lie in the scope of our privacy protection.

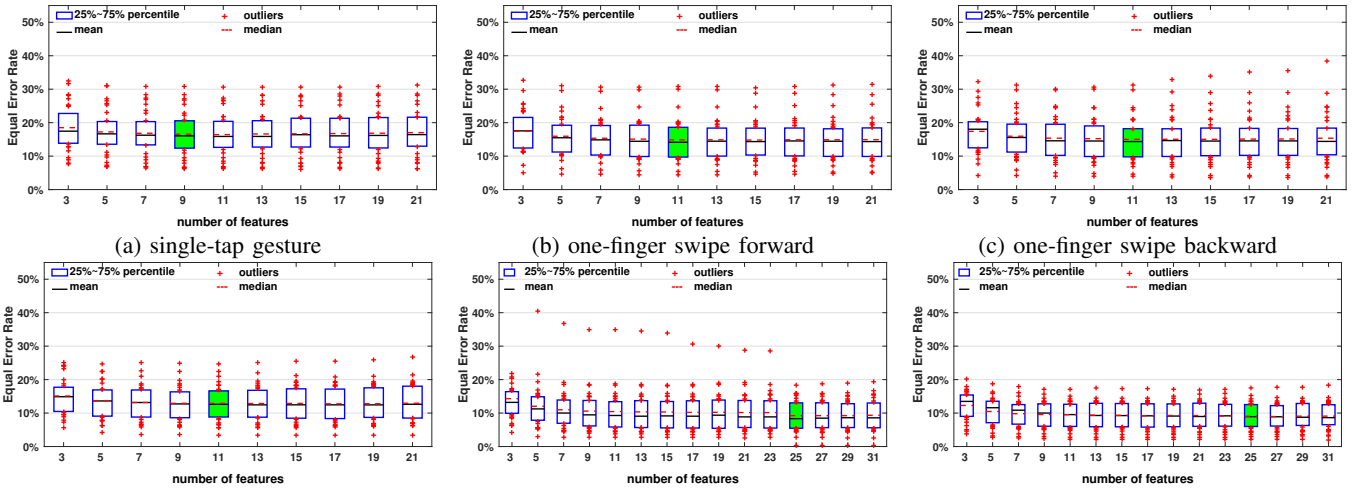
IV. Evaluation

In this section, we evaluate the performance of the GlassGuard authentication system through offline analysis by answering two questions. (1) How well do the classifiers perform? We address this by showing the EER for each classifier in the system. (2) How well does the whole system work? Here, we show the accuracy of all decisions made by the system and the average delay to make a decision. To evaluate the accuracy, we show the detection rate and false alarm rate when only one single type of user event is available, as well as when different types of events are mixed together with equal probability. To evaluate the delay, we show the number of user events needed for the system to make a decision. We also show the accuracy and decision delay under five typical usage scenarios and compare our system with state of the art.

A. Performance of Classification

We use EER as the performance metric for classification. EER is the error rate when FAR is equal to FRR. It can be obtained by intersecting the Receiver Operating Characteristic (ROC) curve with a diagonal of a unit square [38].

Classification of touch gestures. Figure 5 depicts the EERs of the six classifiers for touch gestures. For each classifier, we vary the number of features used and plot the EERs for all 32 users. From all six classifiers, we see that the average EERs decrease at the beginning as the number of features increases. However, as we continue to add more features, the



(a) single-tap gesture (b) one-finger swipe forward (c) one-finger swipe backward

(d) one-finger swipe down (e) two-finger swipe forward (f) two-finger swipe backward

Fig. 5: Equal Error Rate with different numbers of features

improvement of EER is subtle. In some cases, using more features even results in a higher EER. For example, for the swipe backward classifier, the lowest average EER (15.02%) is achieved with 11 features. When 21 features are used, the average EER rises to 15.35%.

When choosing the best number of features to use in the system, we need to consider both the average EER and the maximum EER. We should also balance between accuracy and computation cost. Take the classifier for single-tap gestures as an example. The average EERs with 9 and 11 features are 16.56% and 16.43%, respectively. By adding two more features, the average EER only increases by 0.07%. Taking all these factors into consideration, the best configuration is: 9 features for the single-tap classifier, 11 features for one-finger swipe classifiers, and 25 features for two-finger swipe classifiers. We mark them in Figure 5 with green shade. Later, we use this configuration to evaluate the performance of our GlassGuard system.

Classification of voice commands. The authentication system makes one decision for each audio file, which is recorded from each voice command. With a sliding window, multiple MFCC vectors are extracted from an audio file. And from each MFCC vector, we get a SVM score. If the scores of 80% of the frames in an audio file favor the owner, then the audio file is marked as "true" (from the owner). Otherwise, the audio clip is marked as "false" (from an impostor). Figure 6 shows the EERs of the voice classifier for different users. Although one user has an EER of as high as ∼12%, for most users, the EER is below 5%. The red line shows the average EER, which is 4.88%. These EERs are much lower than those of classifiers for touch gestures.

B. Performance of GlassGuard

To evaluate the performance of the GlassGuard system, we first test the system with only one single type of user event. Then, we mix all types of user events together and test it again. We do grid search [39] to find the best parameters for SVM classifiers. For parameters of the Aggregator, we choose 99% as the expected detection rate and 1% as the expected false alarm rate. As a result, η_1 and η_2 are 99 and 0.0101 respectively, calculated from Equation (9).

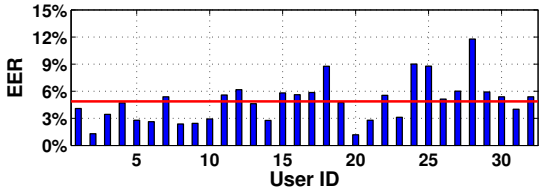


Fig. 6: Classification EERs for voice commands

To do the first test where we only have one single type of user event, we extract all user events of the target type from all users. These events are then used as a user event sequence to feed into our GlassGuard system and test the system performance. Classifications are done in the same way as described in Section III-B. The Aggregator gathers classification results and makes a decision only when it is confident. Every decision is made with events from the same user. If a decision is wrong, we count it as one error. The detection rate of the system is calculated as the ratio of correct decisions with the owner's data. The false alarm rate is calculated as the error rate with impostors' data. To carry the second test, we mix different types of user events together as user input sequences. In addition, we make sure that each event type has the same probability to be chosen for the next user event.

Accuracy. Figure 7 shows the detection rates when different users are taken as the owner. The corresponding false alarm rates are shown in Figure 8. The box shows the 25% and 75% percentiles. The solid line inside the box is the mean value, and the dashed line is the median. First, let us look at the cases when only one single type of user event is available. We see that two-finger touch gestures perform better than one-finger touch gestures. With two-finger touch gestures, all detection rates are above 90% and all false alarm rates are below 5%. With one-finger touch gestures only, both the detection rates and false alarm rates are not as good as those of two-finger touch gestures. A possible reason for this is that two-finger touch gestures have features describing the relative information

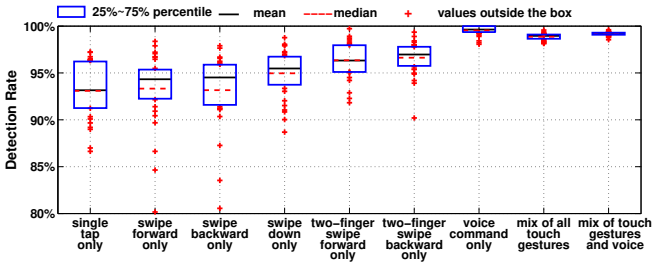


Fig. 7: Detection rate of GlassGuard system

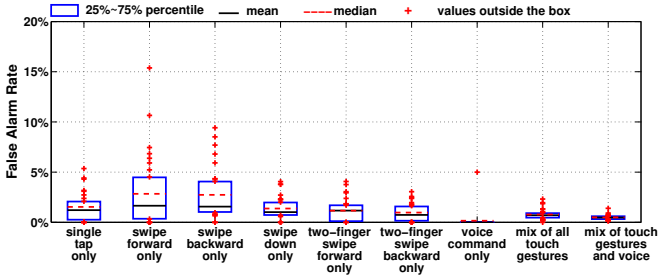


Fig. 8: False alarm rate of GlassGuard system

between two fingers, which are not available in one-finger touch gestures. We also see that in the cases when only a single type of one-finger touch gesture is available, some users have much lower accuracy than others. For example, when only swipe forward gestures are used, user 19 has the lowest detection rate of 81%, and user 5 has the highest false alarm rate of 15%. The deep reason for the lower accuracy of these users needs to be further explored. However, generally, the system works very well. For most of the users, the system achieves a detection rate of more than 90% and a false alarm rate below 10% in all cases. When only voice commands are used, the accuracy is much better than those with any single type of touch gesture. The system has zero false alarm rate with only one exception. In this case, even the lowest detection rate is above 98%. With the low EERs already shown in Figure 6, it is not surprising to see this.

In Figures 7 and 8, we also show the system accuracy when all types of touch gestures are used, and when voice commands are mixed together with touch gestures. The accuracy with all touch gestures is better than any of those individual cases. This is easy to understand as two users may have a similarity in one type of touch gesture but they are different in another one. The mean detection rate in this case is 98.7%, and the mean false alarm rate is 0.8%. With voice commands added, the accuracy is further improved. The mean detection rate increases to 99.2%, and the mean false alarm rate drops to 0.5%. Although they are not as good as those with only voice commands only, they are quite close.

Delay. Figure 9 shows the number of events needed by the system to make a decision when different users are taken as the owner. Similar to the trends of accuracy, when only one type of two-finger touch gesture is used, the average number of touch events needed to make a decision is noticeably less than that when only one type of one-finger touch gesture is used. The case with voice commands only requires the smallest number of events to make a decision, which is below 4 for all users with a mean of 2.24. The number of events needed for

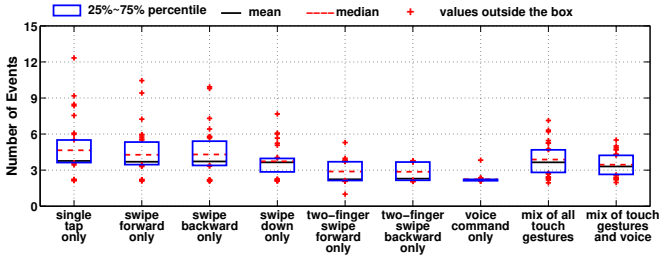


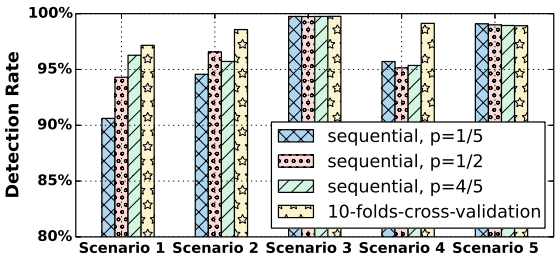
Fig. 9: Number of events needed to make a decision

the case with all touch gestures mixed is in between that of the case with only one type of one-finger touch gesture and that of the case with only one type of two-finger touch gesture. When touch gestures are mixed together with voice commands, the system needs 3.5 user events on average to make a decision.

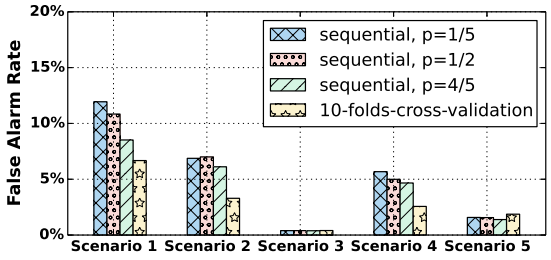
Accuracy and delay in typical usage scenarios. We have demonstrated the performance of our system when only one type of user event is available. We also show the performance when all types of user events are mixed together with equal probability. However, in reality, the distribution of these event types largely depends on what a user wants to do, how the data is organized on the Google Glass, and also a user's own preference (touch gesture or voice command).

Here we take 5 typical usage scenarios and show the accuracy and decision delay under each of the scenarios. (1) Skim through the timeline. A user can access pictures, videos, emails, and application notifications in timeline. Under this scenario, the user swipes forward to see the items in the timeline one by one. The user event sequence consists of only swipe forward gestures. (2) Delete a picture in the timeline. A user does the followings: swipe forward to enter the timeline, continue swipe forward (assume once) to find the group of pictures, single-tap to select the pictures, tap to show the options, swipe forward twice to reach the "Delete" option, and then tap to delete. (3) Take a picture and share it using voice commands. A user event sequence for this is as follows: "OK Glass!", "Take a picture", "OK, glass!", "Share it with...", and swipe down to go back. (4) Take a picture and share it using touch gestures: tap to go to applications, swipe forward (assume twice) to find the picture application, tap to select the application, tap to get the options, tap to select the "Share" option, tap to select a contact, and swipe down to go back. (5) Google search. A user event sequence for this is: tap to go to applications, swipe forward (assume once) to find the picture application, tap to select the application, (speak the keyword), tap to show the options when the content is ready, tap to view the website, two-finger swipe forward to zoom in, and swipe down to return.

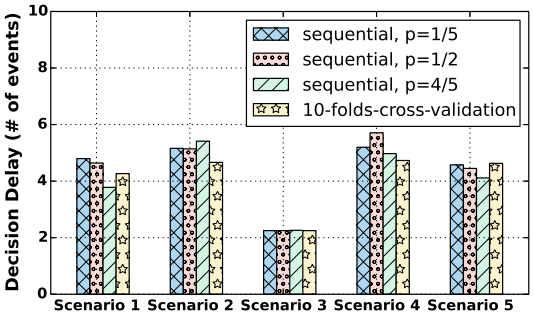
Our aforementioned performance analysis is based on the 10-fold cross validation where training samples are randomly selected. In another word, the training phase happens in parallel with the testing phase. To better indicate the system performance during real deployment, we perform sequential validation where the training phase and testing phase happen in sequence. All N samples are ordered in time sequence. We select the first $p * N$ (p=1/5, 1/2, or 4/5) samples for training and the remaining $(1-p) * N$ samples for testing (except for voice commands of which we have less than 40 samples per



(a) detection rate



(b) false alarm rate



(c) decision delay

Fig. 10: Performance with different training sizes and validation methods under five real usage scenarios

user).

We present the average performance in Figure 10. Form the figure, we have three main observations. First, Scenario 1 has the lowest detection rate as it only contains swipe forward gestures. Scenario 3 mainly consists of voice commands, so it performs the best. Second, in general, larger training size results in better performance. However, the performance gap is small. When $p = 1/5$, we still have detection rate above 90% and false alarm rate below 12%. Third, different validation methods have very similar performance under Scenario 3 because the training for voice commands remain the same.

C. Performance Comparison

As far as we know, the work presented by Chauhan et al. [40] is the only study covering touch behavioral based user authentication on wearable glasses. In their work, the authors also study the performance of touch behavioral biometrics for user authentication on Google Glass. Specifically, they consider four types of touch events: single-tap (T), swipe forward (F), swipe backward (B), and swipe down (D). And they consider seven gesture combinations: T, F, B, D, T+F, T+F+B, T+F+B+D. Different classification models are trained for different gesture combinations. All classification models make predictions independently. To obtain n samples of a gesture combination, each gesture type should appear at least n times. For example, under the Google search scenario (Scenario 5) introduced in the previous subsection, the user event sequence is: TFTVTTFD (where V denotes voice command and F denotes two-finger swipe forward gesture). Their system can get one prediction from the T model with four samples, one prediction from the F model with one sample, and one prediction from the T+F model with one sample. The T+F+B model and the T+F+B+D model, which are able to provide higher accuracy, do not work under this scenario as the swipe backward gesture is not available. Also, the three predictions may be different which makes their system ambiguous. In contrast, our system can freely combine all events within any user sequence and make one final and better decision.

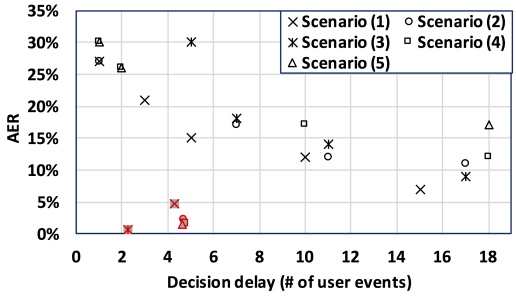


Fig. 11: Comparison of GlassGuard with reference [40]

(Black markers are for the work presented by Chauhan et al. and red markers with shade are for GlassGuard.)

We compare the performance of GlassGuard with the work of Chauhan et al. by showing in Figure 11 the average error rate (AER), which is defined by Chauhan et al. as 1/2*(1 - detection rate + false alarm rate), and decision delay under the five typical usage scenarios introduced in the previous subsection. In the figure, markers in black are for the method described by Chauhan et al., and markers in red are for GlassGuard. Different shapes stand for results under different usage scenarios. For the performance of Chauhan et al.'s work, the lowest AER of all available models is presented. From Figure 11, we see that error rates of Chauhan et al.'s work are above 15% with decision delay of 5 user events. With the same decision delay, our system has much lower error rates. Thus, compared to the work of Chauhan et al., our system achieves better performance.

Also, we notice that, with a certain number of test samples, the highest accuracy of their method is achieved with the model for the T+F+B+D combination. When one of the gesture types is not available, this model is not usable. We have used more samples for training than Chauhan et al. but the difference is not as large as it appears. In their work, 75 samples for the combination of T+F+B+D add up to 300 user events. Besides, even if their accuracy can be improved by increasing the training size, their decision delay does not change because they still need to wait for a fixed number of test samples. And our comparison shows that our system has much shorter decision delay.

V. Related Work

In this section, we articulate how GlassGuard is different from existing works on the topics of continuous and trans-

parent user authentication, user authentication on wearable devices, and other sources for user authentication.

A. Continuous and Transparent Authentication

User authentication has been done via voice recognition [13] and face recognition [41]. However, voice commands are not always available. Asking users to speak from time to time is invasive. Google Glass only has a camera facing away from the wearers. As a result, methods based on face recognition does not work. Moreover, using a camera brings privacy concern.

Early research has studied continuous authentication on personal computers via mouse movements and keystroke dynamics [28], [42], [43]. These two biometrics are much different from touch gestures on wearable glasses.

The idea of using touch behavioral biometrics for user authentication has been validated for multi-touch devices [8], [44]. Since then, various touch behavioral base continuous authentication systems have been proposed. Some of them are based on keystrokes on smartphones [9], [45], [46]. These methods do not work with touch pads on wearable glasses since they do not support keystrokes. Others are based on touch gestures with features extracted from screen touch data[19], [47], [48] and/or features extracted from sensor data during a touch event [10], [18], [49]. However, due to differences in user interaction with wearable glasses and that with smartphones, these authentication systems cannot be directly applied to wearable glasses. The discriminability of those features needs to be evaluated on wearable glasses. Furthermore, users can control wearable glasses with voice commands and easily circumvent the touch-based authentication systems.

Gait information has also been studied [50], [51] for continuous authentication purpose. These works are complimentary to ours as we study the case when users are static.

Conti et al. [52] propose to authenticate a user based on how the user answers or places a phone call, e.g. the movement pattern during the process of bringing the phone to the ear after pressing the "start" button to initiate the call. This method, however, is specific to smartphones. It is not applicable on wearable glasses.

B. User Authentication on Wearable Devices

Physical characteristics of users are explored to do user authentication on wearable devices. Yang et al. [53] measure the difference in user responses to a vibration excitation. This method is intrusive. Cornelius et al. [54] design a new sensor that measures how tissue responds to an electrical current to verify identities of wearers. Similarly, Rasmussen et al. [55] propose to authenticate users based on the human body's response to an electric square pulse signal. These two methods require a specific hardware that is not available in today's smart glasses. Moreover, to apply them in the real world, user safety needs to be addressed.

Chan et al. [56] propose to use the glass camera to scan a QR code displayed on the user's smartphone for authentication. Li et al. [57] propose to authenticate users based on head movements in response to a music cue played on the Google Glass. Both of these options are intrusive.

A similar work to ours is presented by Chauhan et al. [40]. Our comparison in Subsection IV-C shows that our system is more flexible and achieves better performance.

C. Other Sources for User Authentication

Das et al. [58] verify users with questions about the owner's day-to-day experience. This is invasive as users need to answer questions. Usage patterns of smartphone, such as SMS and voice call records, have also been used to do active authentication [59]. This method has long authentication delay as it needs to collect usage data during a long time interval to achieve high accuracy.

Shafagh et al. [60] use information of nearby devices to authenticate a user. Other novel features are also proposed, such as clothes [61] and shoes that a user wears [62]. These methods have potential to be applied in wearable glasses. However, they do not work well alone as a solution for continuous user authentication on wearable glasses because these features are not stable even for the owner. It requires re-training when a user visits a new place or gets new shoes or clothes. However, they can be combined with our system to provide more accurate predictions. Our work is complementary to theirs.

VI. Conclusion and Future Work

In this work, we study a set of touch behavioral features and voice features for user authentication on wearable glasses. With data collected from a user study consisting of 32 participants with Google Glass, the discriminability of these features are then evaluated with SVM models and Sequential Forward Search. With 9 features for single-tap gestures, 11 features for swipe forward/backward/down gestures, and 25 features for two-finger swipe forward/backward gestures, we show that the average EERs of classification based on single type of touch gesture are between 9% and 16.6%. With MFCC vectors extracted from audios, the EER of classification on voice commands is 4.88% on average.

We propose a continuous and noninvasive user authentication system for wearable glasses, named GlassGuard. GlassGuard continuously monitors user touch gestures and voice commands. It employs a mechanism adapted from Threshold Random Walking to make a decision from multiple user events only when it is confident. Our evaluation results based on data collected with Google Glass show that, when decisions are made purely on a single type of user event, the average detection rate is above 93% with a false alarm rate below 3% after less than 5 user events. When all types of user events are mixed with equal probability, our GlassGuard system achieves a detection rate of 99% and a false alarm rate of 0.5% after 3.46 user events. We also demonstrate the performance of GlassGuard with 5 typical usage scenarios, under which the detection rates are above 93.3% and the false alarm rates are below 2.84% after 4.66 events.

In the future, we plan to deploy the proposed system on Google Glass and measure the power consumption. Once the

system is deployed on real devices, we would like to measure the performance under routine daily use by different people other than the five typical ones evaluated in the paper. Also, we plan to validate the applicability of the authentication system over longer term.

REFERENCES

[1] "Smartglasses," https://en.wikipedia.org/wiki/Smartglasses, Accessed: Aug. 2016.

[2] "Consumer & Enterprise Smart Glasses: Opportunities & Forecasts 2015-2020," http://www.juniperresearch.com/researchstore/devices-wearables/smart-glasses/consumer-enterprise-smart-glasses, Accessed Feb. 2016.

[3] D. Shukla, R. Kumar, A. Serwadda, and V. V. Phoha, "Beware, your hands reveal your secrets!" in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 904–917.

[4] N. H. Zakaria, D. Griffiths, S. Brostoff, and J. Yan, "Shoulder surfing defence for recall-based graphical passwords," in *Proceedings of the Seventh Symposium on Usable Privacy and Security*. ACM, 2011, p. 6.

[5] S. Wiedenbeck, J. Waters, L. Sobrado, and J.-C. Birget, "Design and evaluation of a shoulder-surfing resistant graphical password scheme," in *Proceedings of the working conference on Advanced visual interfaces*. ACM, 2006, pp. 177–184.

[6] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, and J. M. Smith, "Smudge attacks on smartphone touch screens." *WOOT*, vol. 10, pp. 1–7, 2010.

[7] S. Schneegass, F. Steimle, A. Bulling, F. Alt, and A. Schmidt, "Smudge-safe: Geometric image transformations for smudge-resistant user authentication," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2014, pp. 775–786.

[8] M. Frank, R. Biedert, E.-D. Ma, I. Martinovic, and D. Song, "Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication," *Information Forensics and Security, IEEE Transactions on*, vol. 8, no. 1, pp. 136–148, 2013.

[9] N. Zheng, K. Bai, H. Huang, and H. Wang, "You are how you touch: User verification on smartphones via tapping behaviors," in *Network Protocols (ICNP), 2014 IEEE 22nd International Conference on*. IEEE, 2014, pp. 221–232.

[10] C. Bo, L. Zhang, X.-Y. Li, Q. Huang, and Y. Wang, "SilentSense: silent user identification via touch and movement behavioral biometrics," in *Proceedings of the 19th annual international conference on Mobile computing & networking*. ACM, 2013, pp. 187–190.

[11] B. Draffin, J. Zhu, and J. Zhang, "Keysens: passive user authentication through micro-behavior modeling of soft keyboard interaction," in *Mobile Computing, Applications, and Services*. Springer, 2014, pp. 184–201.

[12] D. A. Reynolds, "Speaker identification and verification using gaussian mixture speaker models," *Speech communication*, vol. 17, no. 1, pp. 91–108, 1995.

[13] H. Lu, A. B. Brush, B. Priyantha, A. K. Karlson, and J. Liu, "SpeakerSense: energy efficient unobtrusive speaker identification on mobile phones," in *Pervasive Computing*. Springer, 2011, pp. 188–205.

[14] C. Xu, S. Li, G. Liu, Y. Zhang, E. Miluzzo, Y.-F. Chen, J. Li, and B. Firner, "Crowd++: unsupervised speaker count with smartphones," in *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. ACM, 2013, pp. 43–52.

[15] "SiME Smart Glasses," http://www.chipsip.com/archive/SiME%20Smart%20Glasses_2015Jan(1).pdf, Accessed: Aug. 2016.

[16] "Recon Jet," http://www.reconinstruments.com/products/jet/tech-specs/, Accessed: Aug. 2016.

[17] "Vuzix M300 Smart Glasses," https://www.vuzix.com/Products/m300-smart-glasses, Accessed: Aug. 2016.

[18] Z. Sitova, J. Sedenka, Q. Yang, G. Peng, G. Zhou, P. Gasti, and K. Balagani, "HMOG: New Behavioral Biometric Features for Continuous Authentication of Smartphone Users," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 5, pp. 877–892, May 2016.

[19] H. Xu, Y. Zhou, and M. R. Lyu, "Towards continuous and passive authentication via touch biometrics: An experimental study on smartphones," in *Symposium On Usable Privacy and Security, SOUPS*, vol. 14, 2014, pp. 187–198.

[20] D. A. Reynolds, "Experimental evaluation of features for robust speaker identification," *Speech and Audio Processing, IEEE Transactions on*, vol. 2, no. 4, pp. 639–643, 1994.

[21] "Silence removal in speech signals," http://www.mathworks.com/matlabcentral/fileexchange/28826-silence-removal-in-speech-signals, Accessed: Aug. 2016.

[22] "Getevent," https://source.android.com/devices/input/getevent.html, Accessed: Aug. 2016.

[23] "Android Debug Bridge," http://developer.android.com/tools/help/adb.html, Accessed: Aug. 2016.

[24] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *The Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[25] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[26] A. Ben-Hur and J. Weston, "A user's guide to support vector machines," *Data mining techniques for the life sciences*, pp. 223–239, 2010.

[27] B. Schölkopf, K. Tsuda, and J.-P. Vert, *Kernel methods in computational biology*. MIT Press, 2004.

[28] N. Zheng, A. Paloski, and H. Wang, "An efficient user verification system via mouse movements," in *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 2011, pp. 139–150.

[29] W. M. Campbell, J. P. Campbell, D. A. Reynolds, E. Singer, and P. A. Torres-Carrasquillo, "Support vector machines for speaker and language recognition," *Computer Speech & Language*, vol. 20, no. 2, pp. 210–229, 2006.

[30] R. Begg and J. Kamruzzaman, "A machine learning approach for automated recognition of movement patterns using basic, kinetic and kinematic gait data," *Journal of biomechanics*, vol. 38, no. 3, pp. 401–408, 2005.

[31] A. Este, F. Gringoli, and L. Salgarelli, "Support vector machines for TCP traffic classification," *Computer Networks*, vol. 53, no. 14, pp. 2476–2490, 2009.

[32] "LIBSVM Tools," http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/, Accessed: Aug. 2016.

[33] T. Fawcett, "Roc graphs: Notes and practical considerations for researchers," *Machine learning*, vol. 31, pp. 1–38, 2004.

[34] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan, "Fast portscan detection using sequential hypothesis testing," in *Proceedings of IEEE Symposium on Security and Privacy*. IEEE, 2004, pp. 211–225.

[35] G. Gu, J. Zhang, and W. Lee, "Botsniffer: Detecting botnet command and control channels in network traffic," in *Proceedings of the 15th Annual Network and Distributed System Security Symposium, NDSS*, 2008.

[36] M. Xie, H. Yin, and H. Wang, "An effective defense against email spam laundering," in *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 2006, pp. 179–190.

[37] R. LiKamWa, Z. Wang, A. Carroll, F. X. Lin, and L. Zhong, "Draining Our Glass: An Energy and Heat Characterization of Google Glass," in *Proceedings of 5th Asia-Pacific Workshop on Systems*, ser. APSys '14. ACM, 2014, pp. 10:1–10:7.

[38] VLFeat, "Plotting AP and ROC curves," http://www.vlfeat.org/overview/plots-rank.html, Accessed: Aug. 2016.

[39] C. W. Hsu, C. C. Chang, C. J. Lin *et al.*, "A practical guide to support vector classification," Tech. Rep., 2003.

[40] J. Chauhan, H. J. Asghar, M. A. Kaafar, and A. Mahanti, "Gesture-based Continuous Authentication for Wearable Devices: the Google Glass Case," in *International Conference on Applied Cryptography and Network Security*. Springer, 2016, pp. 243–262.

[41] S. Chen, A. Pande, and P. Mohapatra, "Sensor-assisted facial recognition: an enhanced biometric authentication system for smartphones," in *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*. ACM, 2014, pp. 109–122.

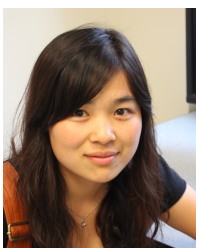
[42] R. Joyce and G. Gupta, "Identity authentication based on keystroke latencies," *Communications of the ACM*, vol. 33, no. 2, pp. 168–176, 1990.

[43] C. Shen, Z. Cai, and X. Guan, "Continuous authentication for mouse dynamics: A pattern-growth approach," in *Dependable Systems and Networks (DSN), 2012 42nd Annual IEEE/IFIP International Conference on*. IEEE, 2012, pp. 1–12.

[44] N. Sae-Bae, K. Ahmed, K. Isbister, and N. Memon, "Biometric-rich gestures: a novel approach to authentication on multi-touch devices," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2012, pp. 977–986.

[45] C. Giuffrida, K. Majdanik, M. Conti, and H. Bos, "I sensed it was you: authenticating mobile users with sensor-enhanced keystroke dynamics,"

in *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2014, pp. 92–111.
[46] B. Draffin, J. Zhu, and J. Zhang, "Keysens: Passive user authentication through micro-behavior modeling of soft keyboard interaction," in *Mobile Computing, Applications, and Services*. Springer, 2013, pp. 184–201.
[47] T. Feng, J. Yang, Z. Yan, E. M. Tapia, and W. Shi, "Tips: Context-aware implicit user identification using touch screen in uncontrolled environments," in *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*. ACM, 2014, p. 9.
[48] L. Li, X. Zhao, and G. Xue, "Unobservable re-authentication for smartphones," in *NDSS*, 2013.
[49] J. Zhu, P. Wu, X. Wang, and J. Zhang, "Sensec: Mobile security through passive sensing," in *Computing, Networking and Communications (ICNC), 2013 International Conference on*. IEEE, 2013, pp. 1128–1133.
[50] J. Mäntyjärvi, M. Lindholm, E. Vildjiounaite, S.-M. Mäkelä, and H. Ailisto, "Identifying users of portable devices from gait pattern with accelerometers," in *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*, vol. 2. IEEE, 2005, pp. ii–973.
[51] C. Nickel, T. Wirtl, and C. Busch, "Authentication of smartphone users based on the way they walk using k-nn algorithm," in *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2012 Eighth International Conference on*. IEEE, 2012, pp. 16–20.
[52] M. Conti, I. Zachia-Zlatea, and B. Crispo, "Mind how you answer me!: transparently authenticating the user of a smartphone when answering or placing a call," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*. ACM, 2011, pp. 249–259.
[53] L. Yang, W. Wang, and Q. Zhang, "VibID: User Identification through Bio-Vibrometry," in *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE, 2016, pp. 1–12.
[54] C. Cornelius, R. Peterson, J. Skinner, R. Halter, and D. Kotz, "A wearable system that knows who wears it," in *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*. ACM, 2014, pp. 55–67.
[55] K. B. Rasmussen, M. Roeschlin, I. Martinovic, and G. Tsudik, "Authentication using pulse-response biometrics," in *Proceedings of the Network and Distributed Systems Security Symposium (NDSS)*, 2014.
[56] P. Chan, T. Halevi, and N. D. Memon, "Glass OTP: Secure and Convenient User Authentication on Google Glass," in *Financial Cryptography Workshops*, vol. 8976. Springer, 2015, pp. 298–308.
[57] S. Li, A. Ashok, Y. Zhang, C. Xu, J. Lindqvist, and M. Gruteser, "Whose move is it anyway? authenticating smart wearable devices using unique head movement patterns," in *PerCom*, 2016.
[58] S. Das, E. Hayashi, and J. I. Hong, "Exploring capturable everyday memory for autobiographical authentication," in *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. ACM, 2013, pp. 211–220.
[59] P. Giura, I. Murynets, R. Piqueras Jover, and Y. Vahlis, "Is it really you?: user identification via adaptive behavior fingerprinting," in *Proceedings of the 4th ACM conference on Data and application security and privacy*. ACM, 2014, pp. 333–344.
[60] H. Shafagh and A. Hithnawi, "Poster: come closer: proximity-based authentication for the internet of things," in *Proceedings of the 20th annual international conference on Mobile computing and networking*. ACM, 2014, pp. 421–424.
[61] S. Richter, C. Holz, and P. Baudisch, "Bootstrapper: recognizing tabletop users by their shoes," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2012, pp. 1249–1252.
[62] H. Wang, X. Bao, R. R. Choudhury, and S. Nelakuditi, "Insight: recognizing humans without face recognition," in *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*. ACM, 2013, p. 7.

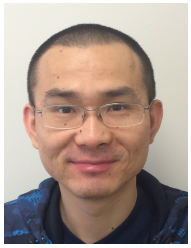


Ge Peng has been a Ph.D student in the Department of Computer Science, College of William & Mary, since 2011 Fall. She received her B.S degree from National University of Defense Technology in 2008. Her research interests include wireless networking, smartphone energy efficiency, and ubiquitous computing.

Gang Zhou Dr. Gang Zhou is an Associate Professor, and also Graduate Director, in the Computer Science Department at the College of William and Mary. He received his Ph.D. degree from the University of Virginia in 2007. He has published over 80 academic papers in the areas of sensors & ubiquitous computing, mobile computing, body sensor networks, internet of things, and wireless networks. The total citations of his papers are more than 5000 according to Google Scholar, among which five of them have been transferred into patents and the MobiSys?04 paper has been cited more than 800 times. He is currently serving in the Journal Editorial Board of IEEE Internet of Things, Elsevier Computer Networks, and Elsevier Smart Health. He received an award for his outstanding service to the IEEE Instrumentation and Measurement Society in 2008. He also won the Best Paper Award of IEEE ICNP 2010. He received NSF CAREER Award in 2013. He received a 2015 Plumeri Award for Faculty Excellence. He is a Senior Member of ACM and also a Senior Member of IEEE.

David T. Nguyen has been working on his Ph.D. in Computer Science at the College of William and Mary, USA, since Fall 2011. He is working with Dr. Gang Zhou, and his research interests include mobile computing, ubiquitous computing, and wireless networking. Before coming to W&M (Fall 2011), he was a lecturer at Suffolk University in Boston for 2 years. He was also a lecturer at Christopher Newport University in 2013. In 2014, he worked as a Mobile Hardware Engineer in Facebook's Connectivity Lab, Menlo Park, CA.

Xin Qi is a Member of Technical Staff III at VMware NSX Group in Palo Alto, CA. He received his Ph.D. degree in Computer Science from the College of William and Mary in May 2015. Before that he received his M.Eng. degree in Pattern Recognition and Intelligent Systems in June 2010 from National Key Laboratory of Pattern Recognition at Institute of Automation, CAS, and his B.Sc. degree in Computer Science in June 2007 from Nanjing University. During his Ph.D., he conducted research in Ubiquitous and Mobile Systems. His current interests are in Ubiquitous Computing, Mobile Systems and Big Data in Cloud.



Qing Yang received his B.S from Civil Aviation University of China in 2003 and M.S. from Chinese Academy of Sciences in 2007. Since Fall 2011, he is a Ph.D student in the Department of Computer Science, College of William & Mary. His research interests are smartphone security and energy efficiency.

Shuangquan Wang received his PhD degree in pattern recognition and intelligent system from Shanghai Jiao Tong University at 2008. Then, he joined Nokia Research Center (Beijing) as a postdoc researcher. From May 2010 to August 2016 he worked at Institute of Computing Technology, Chinese Academy of Sciences (ICT, CAS). Now he is pursuing his second PhD degree at Department of Computer Science, College of William and Mary. His research interests include ubiquitous computing, mobile service and human-centric computing.