Challenge-Response Authentication using In-Air Handwriting Style Verification

Jing Tian, Yu Cao, Wenyuan Xu, Song Wang

Abstract—Challenge-response (CR) is an effective way to authenticate users even if the communication channel is insecure. Traditionally, CR authentication relies on one-way hashes and shared secrets to verify the identities of users. Such a method cannot cope with an insider attack, where a user can obtained the secret (i.e., the response) from a legitimate user. To cope with it, we design a biometric-based CR authentication scheme (hereafter MoCRA), which is derived from the motions as a user operates emerging depth-sensor-based input devices, such as a Leap Motion controller. We envision that to authenticate a user, MoCRA randomly chooses a string (e.g., a few words), and the user has to write the string in the air. Using Leap Motion, MoCRA captures the user's writing movements and then extracts his / her handwriting style. After verifying that what the user writes matches what is asked for, MoCRA leverages a Support Vector Machine (SVM) with co-occurrence matrices to model the handwriting styles and can reliably authenticate users, even if what they write is completely different every time. Evaluated on data from 24 subjects over 7 months, MoCRA managed to verify a user with an average of 1.18% (Equal Error Rate) EER and to reject impostors with 2.45% EER.

Index Terms—Authentication, challenge-response, identification, biometrics, SVM, handwriting style.

1 Introduction

User authentication is one of the most important yet challenging tasks in computer security [?], [?], [?], [?], [?]. The difficulty stems from the insecure communication, where eavesdropping, man-in-the-middle attacks, and replay attacks are all made possible. Challenge-response (CR) authentication can effectively cope with these attacks: typically, during a CR authentication, one party (e.g., a server) sends a random challenge. To be authenticated, the other party (e.g., a user) has to send back a valid response, which is usually the hash of the challenge and the secret shared by the two parties beforehand. Because the challenge is randomly selected and it is difficult to extract the password from the response, CR authentication is considered secure over insecure communication channels. However, such a challenge response method authenticates is based on *what you know* instead of *who you are*. Anyone knowing the shared secret can pass the authentication. Such authentication cannot prevent insider attacks, which are serious threats to systems with strict security requirements. For instance, enterprise or government may only allow the security guards who have passed extensive background checks to patrol their buildings. Letting their friends or colleagues without background checks substitute them is not allowed and can lead to an insider attack. To address it, we study biometric-based challenge-response schemes to authenticate based on *who you are*. Compared with authentication using biometrics, adding the procedure of challenge-response will introduce an extra amount of time to verify the response. Nevertheless, we believe the challenge-response procedure can reduce the possibility of replay attacks.

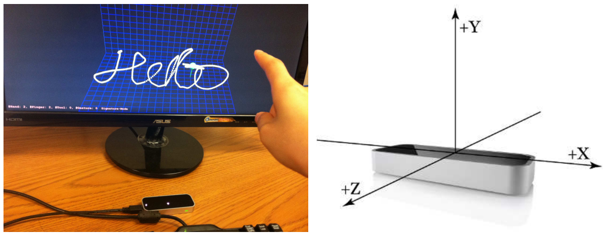


Fig. 1. An illustration of using a Leap Motion controller to acquire a user's handwriting in the 3D space for content independent CR authentication.

Both physical and behavioral biometrics can be used to identify who you are. Considering that physical biometrics neither privacy-friendly,nor variable easily, we focus on designing a behavioral-biometric-based CR authentication system. We envision to design a motion-based authentication system with the following features. First, it works in a contactless manner, which has the benefit to elliminate hygiene concerns and is resilient to smudge attacks [?], i.e., finger smudges (due to oily residues) on a touch screen can reveal passwords. Second, it can be used as a complementary method when traditional biometrics-based authentication is inapplicable. For instance, fingerprints are inapplicable in several scenarios, e.g., fingerprints are unsuitable to users with dirty, greasy, or worn-out fingerprints due to their professions (e.g., miners). Third, it should be applicable to the public scenarios and thus should be resistant to shoulder surfing (i.e., observing) attacks, and acoustic environmental noises. We propose to use in-air handwriting style, which is unique to a user, as a new biometrics for authentication. We use a depth motion sensor, Leap Motion controller [?] to record in-air fingertip writings as shown in Figure 1. Handwriting samples, in the form of scanned images or recorded by tablet, have been used as signature verification or writer identification for authorization or forensics purpose [?]. Compared to writings on tablet, the in-air writings introduce a larger amount of variability which we believe

- *Dept. of Computer Science and Engineering, University of South Carolina, Columbia, SC, 29208.*
E-mails: tian9@cec.sc.edu, cao@cec.sc.edu, wyxu@zju.edu.cn, songwang@cec.sc.edu

Manuscript received XXX XX, 2016; revised XXX XX, 2017.

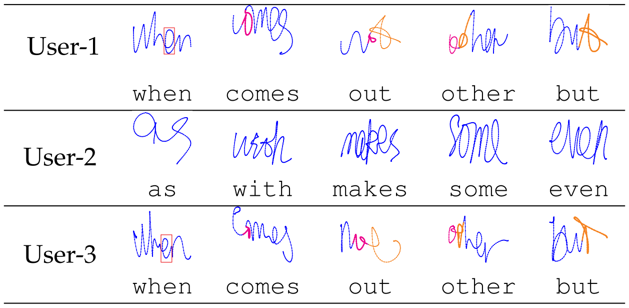


Fig. 2. Examples of processed trajectories written by two users. Despite that some handwritten words from User-1 and User-3 appear to be similar (e.g., `but`) yet some letters written by the same user may differ (e.g., the highlighted letters 'o' and 't'), our authentication system can reliably distinguish them regardless of what they wrote. This is because MoCRA is independent of text contents since it utilizes features derived at the scale of a stroke segment, e.g., a short length of handwritten trajectory.

contain a richer set of biometric yet introduce extra intra-variance for multiple trial of writings from the same writer, and thus impose challenges for authentication. Despite of the challenge, we choose in-air handwriting because it has the following advantages. First, it does not require physical contact to any device. Second, handwriting style is essentially a behavioral biometrics and thus has limited privacy issue. Third, given the rich combination of letters and numbers, the continuously written words are challenging to be synthesized because we believe that imitating arbitrary handwritings in the three dimensional (3D)-space is ambitious, making it resistant to shoulder surfing.

We call the proposed behavioral-biometric-based CR authentication system as MoCRA. Specifically, we use data extracted from finger movements when a user writes in the air. The MoCRA system works as follows. To authenticate a user, MoCRA randomly prompts a string (e.g., a few words) on the screen as a challenge, and the user has to write the string in the air as a response. Then the system performs two steps. First, determine whether the content of the handwriting is the same as the challenge. Second, verify if the handwriting is created by the user. Since handwriting recognition can utilize existing technology [**?**] and recent study shows that Leap Motion has the potential for handwriting recognition application [**?**], [**?**], this paper focuses on the second step — user verification based on the in-air handwriting.

MoCRA has to be *content-independent* for CR authentication, because the CR authentication can require a user to write any content including letters, digits, and special symbols. We design the second step of MoCRA to rely on the handwriting style, i.e., authenticating based on 'how you write' instead of 'what you write'. Building such an authentication system is challenging. First, extracting a handwriting style from 3D movements captured by Leap Motion is non-trivial because of the overlapped finger trajectories created when a user is forced to write multiple words within a limited operation range of Leap Motion (i.e., approximately 25 to 600 mm above the device [**?**]). In addition, MoCRA has to be reliable despite the handwriting variation of the same user (especially writing different content) and possible handwriting similarity between different users (especially writing the same content), and MoCRA has to reject attackers. To characterize handwriting styles, we introduce the concept of *stroke segment*: a segment of fingertip trajectory that is small enough to serve as writing style element.

To make the classification computationally reasonable and better represent the writing style, we propose a transition co-occurrence matrix. We use a *sample*, i.e., a piece of handwriting (e.g., a trajectory recorded in 5 seconds,) to represent a users handwriting style. By combining co-occurrence matrices extracted from data samples and an effective classification method — Support Vector Machine (SVM), we are able to authenticate users reliably. For example, Figure 2 illustrates handwriting of two users. Despite that some handwritten words by User-1 and User-3 appear to be similar (e.g., `but`) yet some letters written by the same user may differ (e.g., the highlighted letters 'o' and 't'), our authentication system can reliably distinguish them regardless of what they wrote. This is because MoCRA is designed to be independent of text contents and utilizes features derived at the scale of a writing stroke segment. Encouragingly, our system can correctly identify all the three users, despite the visual similarity between handwritten words with the same content written by User-1 and User-3. The main contributions of this paper are listed below.

- We designed a motion-based challenge-response authentication that we call MoCRA. MoCRA models handwriting styles in the 3D space and can authenticate users independent on what they write, e.g., English letters, math symbols, diagrams, or digits.
- We proposed a novel 3-level feature extraction method that derives a co-occurrence matrix. Thus we reduce the feature dimension, keep temporal information and statistically represent writing styles.
- We built a system that uses the latest motion sensor, a Leap Motion controller, to capture users' writing movements in the air and performed classification using SVM.
- We evaluated MoCRA on 24 subjects over 7 months and 7 observing attackers. The CR authentication results show that the average equal error rate is 1.18% for random insider attacks and 2.45% for random impostors when testing with 17.5 seconds of handwriting. Besides, we could achieve an average error rate of 3.11% testing with 8.8 seconds of handwriting. In addition, MoCRA can reliably reject observing attackers.

We organize the remainder of the paper as follows. We define the attack model, discuss the background of Leap Motion and related work in Section 2. In Section 3, we discuss the feasibility of authenticating a user using 3D handwriting styles, and show an overview of the MoCRA system. Then, we present the data processing in Section 4, and feature extraction schemes as well as a classification method in Section 5 and show results inSection 6. Finally, we present some discussion and conclude in Section 7.

2 Background and Related Work

This paper aims at designing a biometric-based challenge-response authentication system that can verify legitimate user(s) and reject attackers. Our system utilizes a Leap Motion controller, a 3D motion sensor, to track the user's 3D handwriting, based on which we verify whether what a user write matches what is asked for. Then we perform feature extraction and user authentication.

In this section, we define attack models, introduce the Leap Motion controller, and discuss the related work.

2.1 Background

2.1.1 Attack Model

We assume that the system is used in a well controlled environment and attackers have no physical access to the hardware (i.e., the Leap Motion) nor the software (e.g., the operating systems or databases). Second, the communication path between the Leap Motion and a computer is secure so that no attackers can hijack or inject motion data in between. Attackers can only attempt to impersonate users by writing in front of a Leap Motion controller while mimicking other users. we consider the following two types of attack models.

1) **Non-Observing attackers** try to imitate a legitimate user without any knowledge of the victim's handwriting style, and hope to be identified as the victim with random writings. In particular, Non-Observing attackers can be either an insider that has enrolled in the `MoCRA` system or an impostor without enrollment.

2) **Observing attackers** are better informed and they could have visually observed the victim's writing process, and have viewed finger trajectories of the victim's handwriting displayed on a computer screen.

2.1.2 Leap Motion Controllers

Leap Motion is a motion sensor connected to a computer via a USB port, and it can track the motion of human hands and all ten fingers in the 3D space. Compared to the Microsoft Kinect, the Leap Motion tracks hands including fingers (e.g., finger tips) in a much higher precision but in a smaller space. In particular, it can periodically provide information on finger width, length and motion velocity. Such information reflects the user's hand-motion kinematics, from which we can recognize the user's handwriting style.

Leap Motion is equipped with two cameras and three infrared LEDs [?]. The 3D positions of fingers are derived by combining their 2D positions on the image frames taken by the two cameras and depth measured by the infrared lights. As a user write in front of a Leap Motion (as shown in Figure 1) using one of his finger, a trajectory of the finger is formed by connecting fingertip positions sequentially and is displayed on the screen. Leap Motion uses the Cartesian coordinate system and can track the fingers in a 3D space of an inverted pyramid centered on the device, and the effective range is approximately 25 to 600 mm above the device (1 inch to 2 feet) [?], [?]. Based on our measurements, it can track the position and velocity of the human fingertips at a rate of around 114 frames per second and with a spatial resolution of 0.01mm.

2.2 Related Work

Challenge-response protocols are widely used for user verification over insecure channels. Randomly generatedchallenges and encrypted or hashed responses make the protocols resilient to replay attacks [?] and dictionary attacks [?]. O'Gorman *et al.*, without experimental analysis, briefly suggested to create challenge-response protocols with biometrics [?], including speaker verification, keyboard dynamics. Johnson *et al.* [?] proposed to use voice to construct a challenge response method that can verify users without breaching privacy. Their protocol uses encrypted feature vectors from real users and chaff ones from random people to create a hidden challenge that can only be recognized by the real users. Our work also uses biometrics, but our system utilizes motion instead of voice and can work in a noisy environment.

Recently, gestures embedded in the usage patterns of traditional I/O devices (e.g., keystroke dynamics [?], [?] and mouse movements, or clicks [?], [?]) and new input devices (e.g., wearable accelerometer sensors [?], smart phones [?], [?] and multi-touch screens [?], [?]) can capture different types of gestures for authentication purpose. Our work also try to utilize gestures that are embedded in the usage patterns of input devices. However, none of the prior work studied the gestures associated with the emerging depth sensors, nor can they serve as a basis to construct challenge-response authentication.

Authentication based on depth sensors has been studied on Microsoft Kinect [?], [?] and Leap Motion [?], [?]. Using motion sensors to capture in-air handwriting for authenticatioin was first proposed to enhance text-base passwords [?]. Then, Nigam *et al.* proposed a recognition system based on fusion data of signatures from a Leap Motion sensor and face images from a camera [?]. They both combine the writing content with the behavioral biometrics, and thus requires to write the same content for authentication. Our work is different because we try to harvest the writing style in the 3D-handwriting and do not depend on writing content. Thus, our work represents a harder problem and requires to utilize extra sophisticated features.

Research on handwriting style has been used for identifying the person who wrote a document or determining whether multiple documents are written by the same person. Handwritings could be obtained offline (i.e., scanned images of handwriting [?]), online by a digitizing tablet [?], or in the 3D space [?]. Traditionally, handwriting styles mostly focus on off-line handwritings and features extraction include two classes: textural features (e.g., directionally and curvature of patterns in handwritten images), or allographs extracted from local handwritten patterns (i.e., shapes [?]). To extract handwriting styles, feature study techniques fall into two categories: statistical- and codebook- based feature extraction. For statictical method, Bulacu *et al.* proposed edge based directional probability distributions as features [?]. Schomaker *et al.* proposed joint probability distribution of angle combination of two 'hinged' edge fragments [?] and extended by [?]. The codebook-based features are derived from Bags of (Visual) Words from computer vision community [?]. *Primitives* in the codebook (i.e., *vocabulary*) are local elements that extracted from writing data. Then a histogram of primitives refers from the codebook as characteristic for a user [?]. Schomaker *et al.* used the connected-component contours as the basic elements to capture features of the pen-tip trajectory [?] and then extened to ink-blob shapes [?]. These methods do not necessary work well in our problem because our handwriting is dynamic and contain temporal information (e.g., speed).

On-line handwritings (e.g., handwriting recorded by a tablet) contain temporal information such as the velocity of the pen movements. For content-dependent application, 2D online handwritings are widely used for signature verification [?]. For content-independent applications, the writing style analysis is applied in writer identification. Liwicki *et al.* presented an on-line writer identification system for smart meeting rooms, used features at point (i.e., frame) level and stroke level extracted from text line [?]. Namboodiri *et al.* used low level shape-based features and Li *et al.* used stroke level at probility distribution [?] and then extended to use temporal sequence codes for speed and pressure changes and shape codes for direction [?]. These work analyzed handwriting in 2D space and ours focused on 3D, which we believe introduces additional challenges due to the un-intended issues in the 3D space. In addition, existing writer identification methods used a large amount of testing text for testing, and are not suitable for authentication where userability is the key. For instance, [?] used 80 words for a single test and [?] used one paragraph, about 40 Chinese and/or English characters, for a single test.

Compared to 2D handwriting, 3D handwriting style modeling is challenging, as trajectories are continuously recorded in 3D space, which results in no obvious stroke information or pen-up/down moments. In addition, the touch free input method provides less feedback while writing in-air and thus may result in inconsistency between trials. ?? To address it, we utilize both spatial and temporal information of the in-air writing recordings, adopt the concept of the stroke segment, and extract a vocabulary for writing-style-element representation. Instead of only using histograms of primitives, we introduce a co-occurrence matrix that quantifies the transition information between adjacent stroke segments, and our results show that the co-occurrence matrix method can achieve a better performance than histogram-based methods in our systems.

In particular, MoCRA asks a user to write the content the system provides (the Challenge), and MoCRA has to checks if the input content is the same as the system expected. We call this step as `content matching`. Several literatures can be utilized for the content matching. For instance, MoCRA can utilize online handwriting recognition that has been studied in pattern recognition community for a long time, or similarity comparison based on handwriting recognition. Online handwriting recognition can achieve an accuracy of more than 85% on pure cursive writings or 95% on others [?], [?], [?], [?], while similarity comparison can achieve a higher accuracy, since it does not require the specific recognition of each letter, but a confidence score that shows similarities between two data. Furthermore, recent study [?] achieved a recognition accuracy of 97.59% for in-air English character recognition, with data from two depth sensors, Kinect and Leap Motion Controller. These methods could be applied to the content checking step of the challenge-response mechanism.

3 MoCRA System Overview

3.1 Why does MoCRA work?

The MoCRA system consists of two steps: verifying whether what a user writes matches what is asked for, and verifies the identity of the user. The first step is not the focus of this paper and can be accomplished by utilizing the prior work explained in Section 2.2. In particular, the user writes the content that the system provides (the Challenge). After receiving the input from the user (the Response), the system checks if the input content is the same as the system expected. As such, an attacker cannot simply 'replay' handwriting performed in the past by a legitimate user. Based on the assumption that the replay attack would not be an issue for MoCRA, we focus on the second step: how to verify users based on their handwriting styles, i.e., based on *how* they write instead of *what* they write. The challenges are to correctly recognize a user even if he/she writes different contents, and to distinguish users even if they write the same content. The difficulties stem from the possibility of handwriting variation (especially in writing different contents) of the same user and occasional handwriting similarity (especially in writing the same content) between different

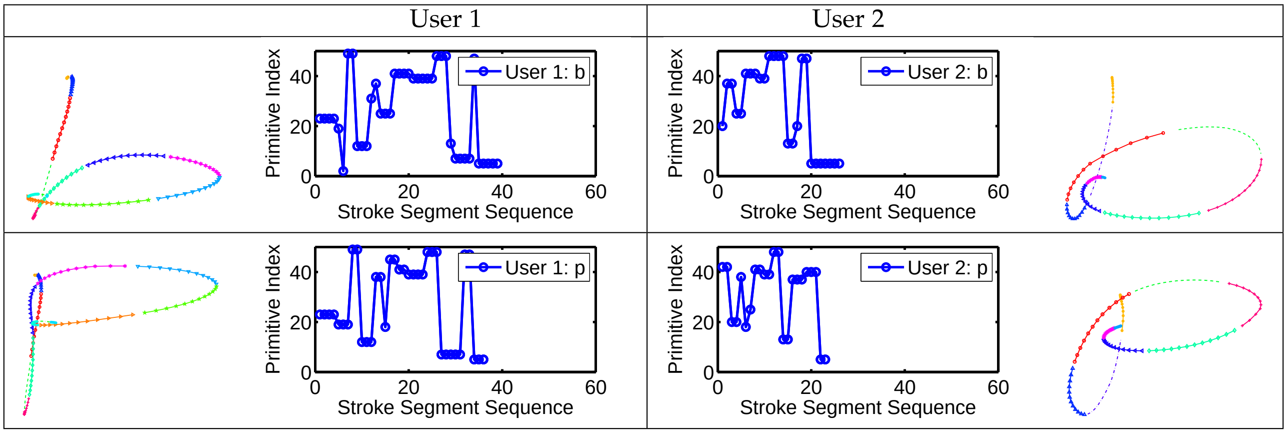


Fig. 3. An illustration that characters written by the same user exhibit similarity while the ones by different users exhibit difference. We use 50 types of stroke segments (i.e., primitives) that consists of 12 continuous frames for illustration. The motion trajectories are divided into stroke segments, and the plots show the primitive-index (y-axis) change over time (x-axis). Plots with similar profiles represent the similarities of the handwriting style. Stroke segments (denoted by different colors on the characters) of `b` and `p` from the same user (either User 1 or User 2) show similar patterns, i.e., similar sequences of primitive indices, but the stroke segment sequences from the different users of the same letter (e.g., `b` or `p`) show distinguished patterns.

users. The key to overcome the challenges is to characterize handwriting styles effectively and efficiently.

Stroke Segments for Effective Modeling. The model characterizing handwriting styles has to be content-independent. A naive approach could be to extract fingertip trajectories that represent each individual character and then to group the ones of the same characters for further comparison. However, such a method may be overkill, as the handwriting in the 3D space is difficult to be delimited precisely, as shown in Figure 2. Although content recognition is possible, perfectly delimiting the fingertip trajectory of each letter is challenging. To avoid the burden of extracting individual symbols, we choose to characterize handwriting styles by short-length continuous trajectories, which are analogous to strokes [?], [?]. In practice, it is difficult to accurately divide a handwriting trajectory into meaningful strokes with variable lengths. We simply divide fingertip trajectories into a set of short, fixed-length *stroke segments*. To reduce the impact of the starting point on a trajectory for stroke segment partition, we apply a temporal-sliding window over the fingertip trajectory for constructing stroke segment, and the constructed stroke segments can be partially overlapped.

Stroke segments can be considered as the basic building blocks that compose symbols. Although the underlying content in fingertip trajectories may be different, some characters may share similar stroke segments. Thus, the two fingertip trajectories created by the same user when writing different sets of words could contain a large percentage of the similar stroke segments. The stroke segments belonging to different users typically show little similarity. For instance, as shown in Figure 3, the letters `b` and `p` have similar composition. Some stroke segments (denoted by different colors on the characters) of `b` and `p` from the same user show similarity, but the Stroke segments of User 1 are different from the ones of User 2. Thus, it is imaginable that the trajectories of the word `'bob'` and `'pop'` from the same user may have many similar stroke segments, but the ones from different users may share few similar stroke segments.

Vocabulary for Improving Efficiency. We define a *frame* on a trajectory as a fingertip position, and consider the associated coordinates and kinematic features at each frame: *frame-level features*. To compare the similarity between stroke segments, we can concatenate the frame-level features of all frames of a stroke segment to form *stroke segment-level features*. Then, we can combine all the stroke segment-level features of a trajectory sample to construct one *style-level feature*, which is the unit to represent the handwriting style. However, simply combining all the stroke segment-level features will lead to high-dimensional vectors. To address this issue, we can define a stroke segment vocabulary that best represents the collection of stroke segments of all the enrolled users. The vocabulary consists of a set of primitives corresponding to typical stroke segments of those users, and each primitive will be assigned a unique index. The creation of a vocabulary can be conducted during the training phase. During the testing stage, each stroke segment is assigned the index of its nearest primitive. This way we reduce each high-dimensional stroke segment-level features to an integer index.

Finally, to compare the similarity between handwriting

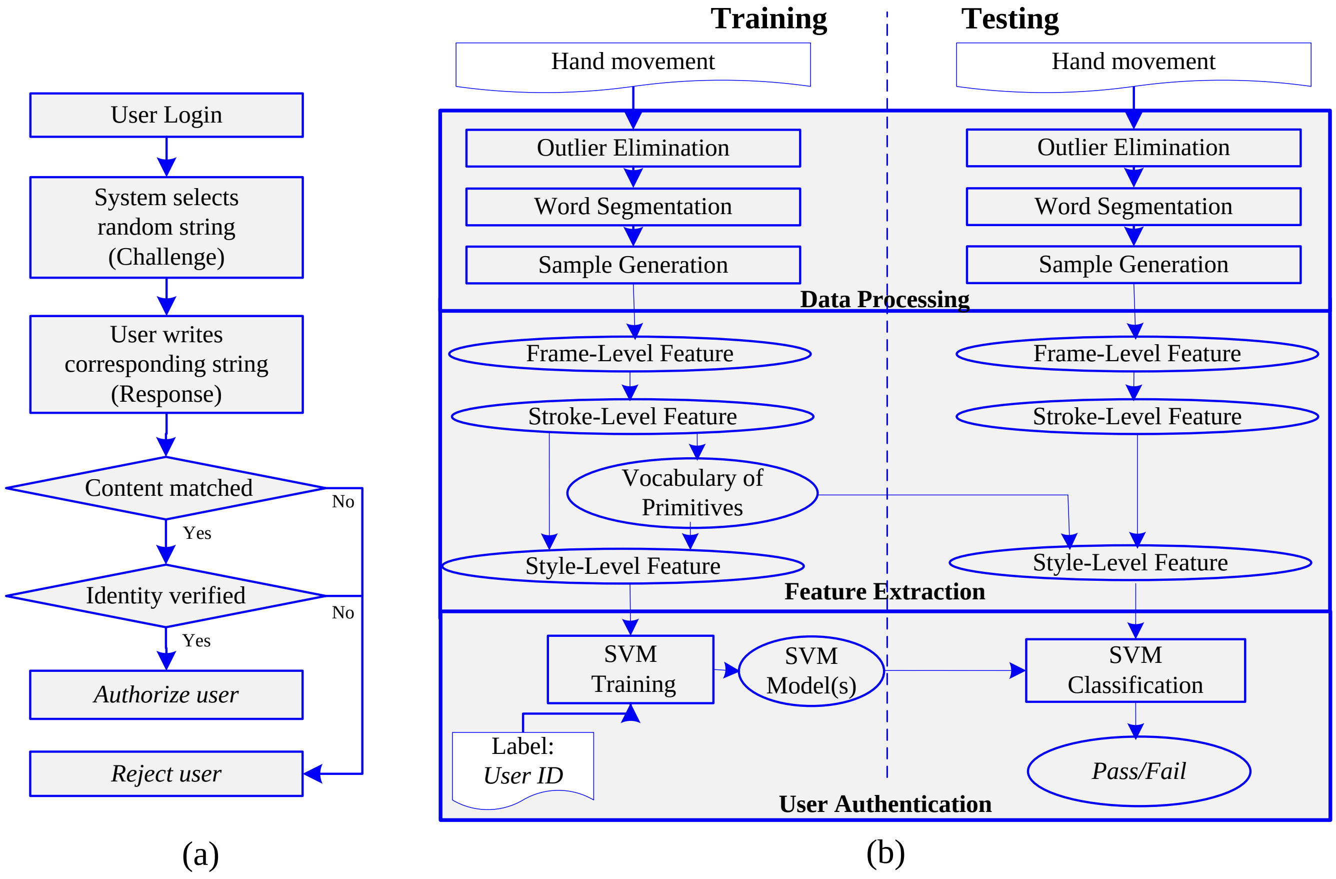


Fig. 4. (a) Flow chart of `MoCRA` system. (b) Flow chart of identity matching. Details of the 3-Level feature extraction are shown in Figure 8.

styles of multiple samples, we perform statistical analysis to achieve content-independence. Specifically, we obtain statistics on the stroke segment indices within a trajectory sample for constructing a low-dimensional feature for a sample. Instead of using the histogram or probability density functions (PDFs) of the individual stroke segment indices [?], [?], [?], we examine the temporal transition between stroke segments. The intuition is that a user may tend to write the same sequence of stroke segments, and such a sequence may be essential to represent handwriting styles. Concretely, we construct a co-occurrence matrix that counts the number of occurrences of each possible stroke segment (index) transition between temporally adjacent stroke segment pairs. This co-occurrence matrix reflects the distribution of the temporal stroke segment transition and we reshape it into a vector as a feature vector of a sample.

Continue with the example in Figure 3, 50 primitives were derived to form a vocabulary for illustration. We observe that the similar letter sets (e.g., `b` and `p`) written by the same user contain similar stroke segment indices and transition pattern of stroke segment indices, while the stroke segment sequences of the same letter (e.g., `b` or `p`) written by different users share few similarities. This example encourages us to study the effectiveness of using vocabulary and transition between stroke segments to model the handwriting style.

3.2 How does MoCRA work?

The MoCRA system consists of a Leap Motion for capturing fingertip movements in the 3D space, computing and storage units for challenge-response processing and identity authentication. Figure 4(a) shows a the flow chart of the challenge-response authentication.

The MoCRA authentication process consists of two phases: enrollment and testing. During an enrollment, the system will capture the initial handwriting and create an account for a user. These handwriting inputs will be used for training. In a testing phase, the system first select a random string. After capturing the user's writing movements, the system first performs a content check, i.e., verifying whether what a user writes matches the random string, and then tests the user's identity. Figure 4 illustrates a detailed flow chart of identity matching.

Content Matching. The focus of this paper is to study the biometric built on handwriting motion instead of content check, because several literatures can achieve the content check, as shown in Section 2.2. Therefore, we do not include the technique details of this part in the paper.

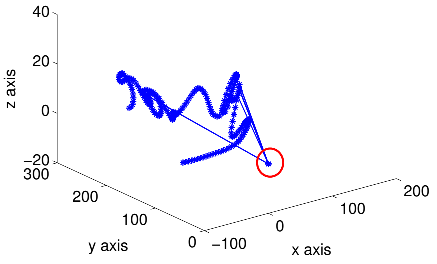


Fig. 5. An illustration of the outliers in Leap Motion data. The red circles highlight the frames with outliers.

Data Processing. Both training and testing phases require data processing and feature extraction. The goal of data processing is to prepare the raw motion data captured by Leap Motion and generate a handwriting sample, which consists of a number of frames that represent the motion of the fingertip. The data processing will remove outliers and meaningless transition trajectories from the data, as well as partition the handwriting trajectory into samples, on which the features can be calculated to represent the writing style.

Feature Extraction. After data processing, a MoCRA system extracts three levels of features from each sample: frame-level features, stroke segment-level features, and style-level features. Level by level, the MoCRA system is able to derive features that effectively and efficiently model the handwriting styles of users.

SVM Training and Testing. In Figure 4, MoCRA system uses the extracted style-level features for both classifier training and testing. The classifier has to achieve two goals: correctly authenticate a legitimate user and reject any impostors that are not part of the pool.

4 Data Processing

Data processing constructs samples from raw data recorded by Leap Motion, and consists of *outlier elimination*, *word segmentation*, and *sample generation*.

4.1 Leap Motion Data

Leap Motion captures finger movements in a 3D space (as shown in Figure 1): the left-right motion will be recorded at the x-direction, the up-down motion at y-direction and the forward-backward motion at the z-direction. As a user raises his/her hands and uses one of the fingers to write, a Leap Motion controller will capture information of up to 10 fingers (depending on their visibility). We extract the data of the foremost fingertip (i.e., the ones with the smallest z-value among all captured figures), and record a 11-dimensional vector with the information provided directly by Leap Motion (listed below) at time t as a *frame*.

1) fingertip position, $-(p_x(t), p_y(t), p_z(t))$,
2) fingertip velocity, $-(v_x(t), v_y(t), v_z(t))$,
3) fingertip direction, $-(D_x(t), D_y(t), D_z(t))$,
4) finger visible length and width. $-$ $L_f(t)$ and $W_f(t)$.

In addition, Leap Motion provides an ID number and a timestamp for each frame. The ID number of each frame for objects (e.g., fingers) is given as a positive number if at least one object is detected or becomes a negative number if no recognizable objects are detected. Thus, we utilize the ID numbers to check the validity of a frame and utilize timestamps to calculate the speed of handwriting. For each round of recording, Leap Motion will record a collection of frames, by sequentially connecting all consecutive frames, we construct a raw handwriting trajectory.

4.2 Outlier Elimination

Noises caused by environment variation may affect the Leap Motion raw trajectories. In general, such noises lead to abrupt changes of the fingertip positions between adjacent

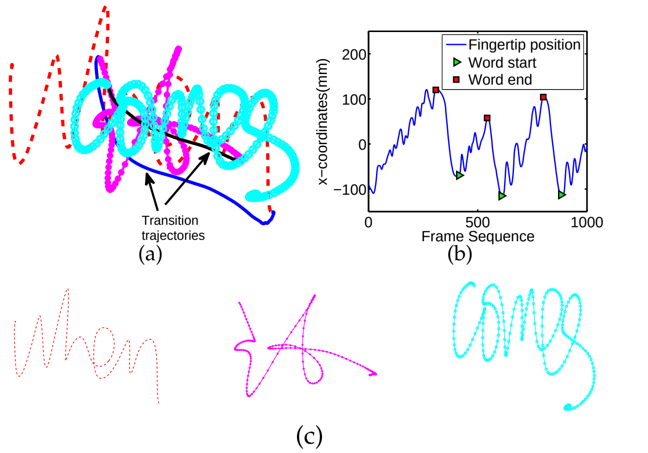


Fig. 6. An illustration of word segmentation. (a) what was recorded by the Leap Motion controller, which contains three words and transition trajectories connecting them. (b) word segmentation results on x-axis. The trajectories starting from a red square and ending at the next green square are the transition ones, which are removed to obtain word segments. (c) shows the separated words from the trajectory in (a). Note that the subject draw the point before the lower part of the letter 'i'.

frames, as shown in Figure 5. Therefore, they are actually outliers. In general, we observe two types of outliers: (1) no fingertip is detected (7.32% amount of data) , and (2) a fingertip is detected but at an unlikely position (less than 1% amount of data). For no-detection cases, we examine the number of consecutive frames that do not contain detected fingertips. If the number is small, we perform spatial interpolation. If the number of noisy frames is large, we discard the noisy frames and divide the raw trajectory into two. Then we can perform the remaining data processing and feature extraction steps on each trajectory independently. For unlikely-position cases, we apply a temporal median filter to remove the outliers [?]. At each frame, we examine a temporal window centered at this frame, search for the median of all values in this window, and set the median as the new value of this frame. The median filter is applied not only on the x, y and z positions but also values on other dimensions. For both cases, our algorithm only removes the outliers and preserves the smoothness and continuity of the handwritings.

4.3 Word Segmentation

The main goal of word segmentation is to identify the segments of the handwriting trajectories that contain useful information for recognizing handwriting styles. When writing on paper, we proceed from left to right without text overwriting. However, Leap Motion has limited writing space within which the hand motion can be captured – after writing one or two words, a finger becomes out of the Leap Motion's field of view and it has to be moved back to the left. As a result, finger trajectories of multiple words are overlapped and connected with transition trajectories that were traditionally invisible on paper, as shown in Figure 6 (a).

The transition trajectories shall be removed to extract real words. The key to identify such transition trajectories is to analyze the variations of x, y and z-coordinates of the handwriting trajectories (after the noise elimination). As a user writes from left to right, the x-coordinates increases gradually. In comparison, re-positioning the hand back to the left bottom corner results in a sudden and large decrease of the x-value, as shown in Figure 6(b), where the x-value periodically decreases since the user has to move the fingertip back to the left end frequently. Thus, we identify transition trajectories by searching for a sequence of frames along which the x-value of fingertip positions monotonically decreases in a short period of time. By removing transition trajectories (e.g., starting from a red square and ending at the next green triangle as shown in 6(b)), we obtain a set of disjoint word segments shown in 6(c)).

MoCRA sets hotkeys for starting a recording and stopping a recording. Before starting, finger(s) stay motionless and start writing after the press of the start hotkey. MoCRA introduces a few data frames before the intent writing and includes these frames in the first word of a recording. The recording will be ended if any of the two scenarios happen: the finger(s) are out of the recording area or the stop hotkey is pressed. Thus, it is possible that extra frames are added to the last word before the recording ends completely.

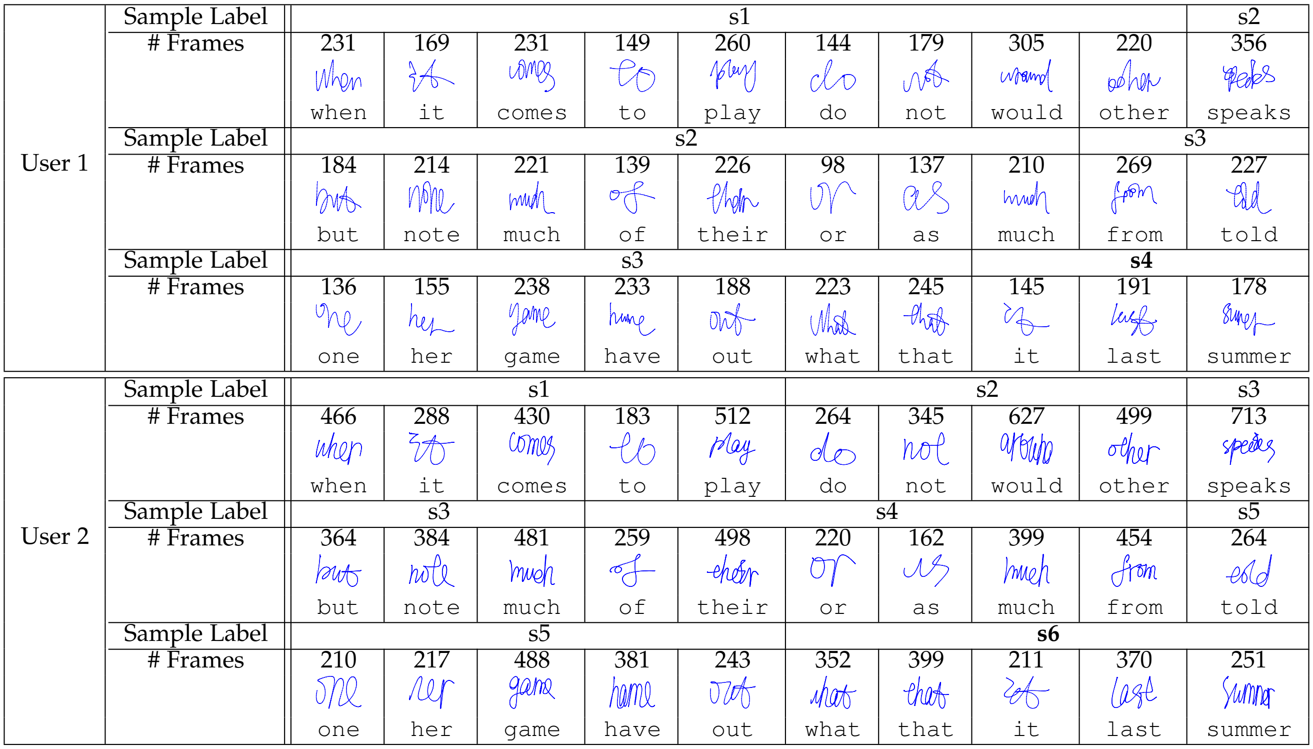
4.4 Sample Generation

The goal of sample generation is to create a set of samples. Since a single word may be too short to represent a user's handwriting style, we construct one sample with multiple word segments. We denote the total number of frames in a sample to be the *length* of the sample. Ideally, the longer the sample length, the better the verification/identification performance. In practice, the length of samples should be small to ensure satisfying usability. In this paper, we conducted a series of experiments to understand the trade-off between usability and security, and determine the appropriate length of samples.

Since various users may write at different speeds, the number of frames in each word segment of various users may differ. To avoid the impact of writing speeds, we construct a sample based on the frame length instead of the number of words. Figure 7 demonstrates an example. In this example, we construct samples with a sample length no greater than 2000 frames. We concatenate as many word segments as possible into a sample until one extra segment will make the sample contain more than 2000 frames. With 30 word segments, user 1 can only create less than 4 samples, and user 2 can create almost 6 samples.

5 Feature Extraction and Classification

In this section, we elaborate how to extract features at the frame-, stroke segment- and style-levels, respectively (shown in Figure 8). Given a sample, we extract frame-level features (denoted by f_f^i for the i-th frame) and then combine the features of L_{ss} consecutive frames to generate a stroke segment-level feature (denoted by f_{ss}^i for the i-th stroke segment). After finding the index of the primitive that is closest to each stroke segment, we obtain a sequence of indices. Then, the occurrences of stroke segment-transition pairs are counted for creating a style-level feature. Then, we discuss the SVM classification for verifying users.



User 1	Sample Label	s1									s2
	# Frames	231	169	231	149	260	144	179	305	220	356
		when	it	comes	to	play	do	not	would	other	speaks
	Sample Label	s2								s3	
	# Frames	184	214	221	139	226	98	137	210	269	227
		but	note	much	of	their	or	as	much	from	told
	Sample Label	s3							**s4**		
	# Frames	136	155	238	233	188	223	245	145	191	178
		one	her	game	have	out	what	that	it	last	summer
User 2	Sample Label	s1					s2				s3
	# Frames	466	288	430	183	512	264	345	627	499	713
		when	it	comes	to	play	do	not	would	other	speaks
	Sample Label	s3			s4						s5
	# Frames	364	384	481	259	498	220	162	399	454	264
		but	note	much	of	their	or	as	much	from	told
	Sample Label	s5					**s6**				
	# Frames	210	217	488	381	243	352	399	211	370	251
		one	her	game	have	out	what	that	it	last	summer

Fig. 7. An illustration of constructing samples from a group of words. All the word segments with the same sample label (e.g., s1, s2) constitute a sample. Given the sample length L_s that is no larger than 2000 frames, 30 words form around 4 or 6 samples for Users 1 and 2, respectively, due to the different writing speeds.

5.1 Frame-Level Feature

For a frame t, we construct a 19-dimensional feature vector, which comes from eight types of kinematics features, as summarized in Table 1. Out of 19 dimensions, 11 are provided by Leap Motion directly, and 8 are calculated.

1) **Position**. The 3D fingertip position in the t-th frame is denoted as $\mathbf{p}(t) = (p_x(t), p_y(t), p_z(t))^T$. To make the positions among stroke segments comparable, all the fingertip positions within a stroke segment are normalized by subtracting the fingertip position in the first frame of the stroke segment. Thus, the fingertip position in the first frame of each stroke segment is always $(0, 0, 0)^T$.

2) **Position Difference between Frames**. The inter-frame position difference is defined as $d(t) = \|\mathbf{p}(t+1) - \mathbf{p}(t)\|$.

3) **Velocity**. The velocity of the fingertip motion in the t-th frame is provided by the Leap Motion and we denote it as $\mathbf{v}(t) = (v_x(t), v_y(t), v_z(t))^T$.

4) **Acceleration**. The acceleration of the fingertip motion in the t-th frame is derived from the velocity, as $\dot{\mathbf{v}}(t)$.

5) **Direction**. The direction of the finger in the t-th frame is provided by the Leap Motion and we denote it as $\mathbf{D}(t) =$

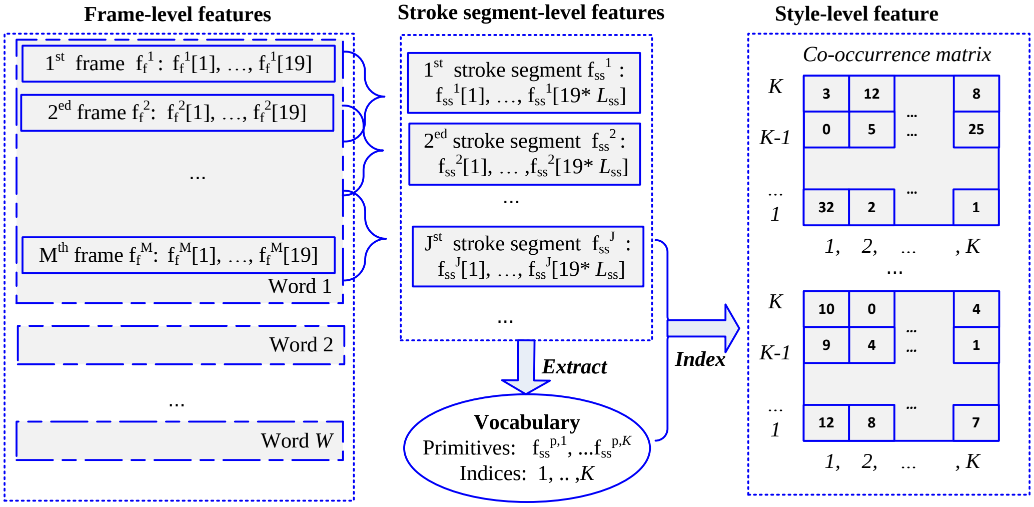

Frame-level features
1st frame f_f^1: f_f^1[1], …, f_f^1[19]
2ed frame f_f^2: f_f^2[1], …, f_f^2[19]
…
Mth frame f_f^M: f_f^M[1], …, f_f^M[19]
Word 1
Word 2
…
Word W
Stroke segment-level features
1st stroke segment f_ss^1 : f_ss^1[1], …, f_ss^1[19* L_ss]
2ed stroke segment f_ss^2 : f_ss^2[1], … ,f_ss^2[19* L_ss]
…
Jst stroke segment f_ss^J : f_ss^J[1], …, f_ss^J[19* L_ss]
…
Extract
Vocabulary
Primitives: f_ss^p,1, …f_ss^p,K
Indices: 1, .. ,K
Index
Style-level feature
Co-occurrence matrix
K
K-1
…
1
3 12 … 8
0 5 … 25
32 2 … 1
1, 2, … , K
…
K
K-1
…
1
10 0 … 4
9 4 … 1
12 8 … 7
1, 2, … , K

Fig. 8. An illustration of constructing a style-level feature, i.e., a co-occurrence matrix. The rectangular with dash lines represent various levels of features and an ellipse with solid lines represent 'vocabulary'.

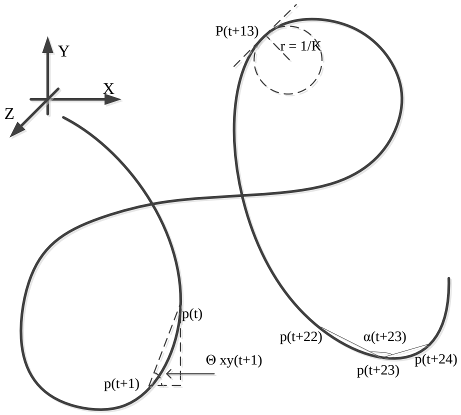


Fig. 9. An illustration of slope angle, path angle and curvature.

$(D_x(t), D_y(t), D_z(t))^T$.

6) **Finger Size**. The visible length and width of the finger in the t-th frame are recorded and provided by Leap Motion and we denote them as $L_f(t)$ and $W_f(t)$, respectively.

7) **Slope Angle**. The slope angles at the t-th frame are defined as

$$\theta_{xy}(t) = \arctan \frac{\dot{p}_y(t)}{\dot{p}_x(t)},$$
$$\theta_{zx}(t) = \arctan \frac{\dot{p}_x(t)}{\dot{p}_z(t)}.$$

8) **Path Angle**. Path angle $\alpha(t)$ is the angle between lines $\mathbf{p}(t)\mathbf{p}(t+1)$ and $\mathbf{p}(t-1)\mathbf{p}(t)$, as shown in Figure 9.

9) **Curvature**. As show in Figure 9, we calculate the log radius of curvature, $\log \frac{1}{\kappa(t)}$, at the t-th frame as one feature, where $\kappa(t)$ is the curvature:

$$\kappa(t) = \frac{\sqrt{c_{zy}^2(t) + c_{xz}^2(t) + c_{yx}^2(t)}}{(\dot{p}(t)_x^2 + \dot{p}_y^2(t) + \dot{p}_z^2(t))^{3/2}},$$

and

$$c_{zy}(t) = \ddot{p}_z(t) \times \dot{p}_y(t) - \ddot{p}_y(t) \times \dot{p}_z(t).$$

We normalize the feature data in each dimension such that it conforms to a standard Gaussian distribution in each word. Specifically, we first calculate a mean value and its standard deviation. Then the values are converted to new values that fit into a standard Gaussian distribution.

5.2 Stroke segment-level Feature

As mentioned above, we construct short, partially overlapped, and fixed-length stroke segments by dividing the word segments of a sample. Let the length of a stroke segment be L_{ss}, and the length of a word segment be L_w. Define the overlapped ratio r to be the number of shared frames between a pair of adjacent stroke segments divided by the stroke segment length L_{ss}. Then the i-th stroke segment starts at the frame of $(1-r)L_{ss} * i$ and its length is L_{ss}. This way, we can construct around $\left\lfloor \frac{L_w}{(1-r)L_{ss}} \right\rfloor$ stroke segment for a word segment. Given a sample that consists of n word segments with length $L_w^i, i = 1, 2, \ldots, n$, respectively, we can in total construct $\sum_{i=1}^{n} \left\lfloor \frac{L_w^i}{(1-r)L_{ss}} \right\rfloor$ stroke segments.

TABLE 1
Features extracted in the frame level.

Type	Features
Positions & Distance	$\mathbf{p}(t)$, $d(t)$
Velocity & Acceleration	$\mathbf{v}(t)$, $\dot{\mathbf{v}}(t)$
Direction	$\mathbf{D}(t)$
Finger Size	$L_f(t)$, $W_f(t)$
Slope angle & Path angle	$\theta_{xy}(t)$, $\theta_{zx}(t)$ and $\alpha(t)$
Log radius of curvature	$\log \frac{1}{\kappa(t)}$

To combine the frame-level features in a stroke segment into a stroke segment-level feature, we sequentially concatenate the features extracted from all the frames. With 19-dimension feature at frame level, the dimension of this stroke segment-level feature is $19 * L_{ss}$.

5.3 Style-level Feature

Simply concatenating all stroke segment-level features of a sample together will create a style feature vector of huge dimensions. For instance, for a sample with n word segments, the dimension of the style feature vector is

$$\sum_{i=1}^{n} \left\lfloor \frac{L_w^i}{(1-r)L_{ss}} \right\rfloor * 19.$$

To reduce the dimension of style-level features, we first quantize the stroke segment-level features. We use training samples to learn stroke segment primitives. Specifically, for each sample, we extract all the stroke segments from all the training samples. We then use K-means algorithm to group the stroke segment-level features into K clusters. The center of each cluster is considered as a primitive of stroke segment-level features. With the K clusters, we achieve a vocabulary with K primitives. We index these K primitives consecutively from 1 to K. With these primitives, we can quantize any stroke segment-level feature, for both the training samples and the testing samples, by finding its nearest primitive and assigning the stroke segment with the index of the corresponding primitive.

We then construct the style-level feature by examining the transition of consecutive stroke segments. Specifically, for each pair of sequential stroke segments (with an offset of $(1-r)L_{ss}$ frames in the same word segment), we denote their transition as an ordered pair $< i, j >$, where i and j are the primitive indices of these two stroke segments. Scanning all such stroke segments pairs in a sample, we can build a $K \times K$ co-occurrence matrix, in which the ij-th element indicates the number of $< i, j >$ stroke segment transitions in this sample. We finally reshape this matrix into a K^2 dimension vector as the feature of this sample. Figure 10 shows examples of the constructed style-level features in the form of the co-occurrence matrix, from two users' samples.

5.4 Classification

We choose SVM [?] as the classification algorithm, because SVM is relatively efficient and showed accurate classification results in many real-life systems. SVM utilizes a "kernal trick" to generalize data well even for high-dimension features. In our experiments, we choose the (Gaussian) radial basis function (RBF) kernel. We use grid search method

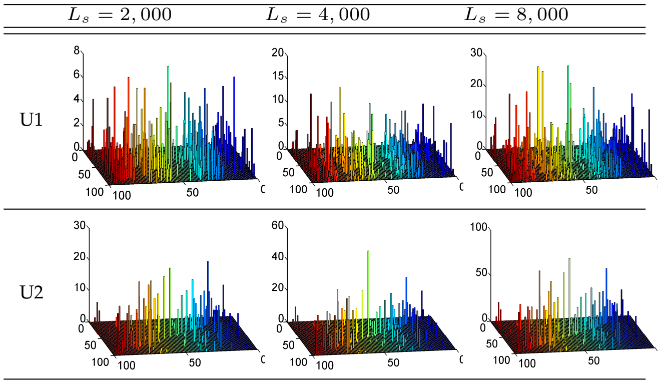


Fig. 10. style-level features (co-occurrence matrix) extracted from samples of two users. We choose $K = 100$ for illustration purpose. As the sample length increases, the shape of co-occurrence matrices remain similar, but the value for each element (i.e. the occurences) increases. The results suggest that the matrices can model handwriting styles.

to optimize RBF SVM parameters C and γ [?], and other parameter are set as default . Finally, we set γ as 0.01 and the penalty parameter C as 5000.

For challenge-response authentication, the main goal of the classifier is to verify a user's identity, thus we train multiple binary-class classifiers for all users. We split the input data into two parts: training data and testing data. The training data are used to model the writing style of a user, and a testing sample is used to test a user's writing style, thus a sample should include writing style representations of the user. Since the writing styles are extracted from stroke segments instead of letters. Thus, as long as enough stroke segments are used for training and testing samples, MoCRA can model a user's writing style reliably. We assume the content of the challenge has a fair distribution, so the length of a sample (i.e., how many letters, stroke segments or frames in a sample) indirectly shows the representations of the writing style. We will examine appropriate length in Section 6.1.2.

For each user, we take training samples from this user as positive samples and the other users' training samples as negative samples and train a binary SVM classifier. Given a new test sample and the user identity it claims, we test it with the user's binary SVM classifier. The sample will be authorized if the classifier returns a positive response, and be rejected with a negative result. In addition, the classifiers should reject any impostors, the samples of which are never part of the training data.

6 Experiment and Evaluation

We evaluate the performance based on the data collected from 24 subjects, who are mainly graduate students between 25 to 35-year old. We choose a 19-paragraph, 830-word article from *New York Times* and then ask each subject to write the whole article once — in front of a Leap Motion sensor from time to time over a period of 7 months. The word length varies from 1 to 14 characters.

Writing in the air is prone to cause arm fatigue [?]. Since the least physically demanding position keeps the upper-arm at rest [?], we let the subjects lay their elbows on a table to rest their upper-arms, and bend their elbows to further reduce fatigue. The larger the angle between the table and the subjects' forearms, the less fatigue they experienced. In addition, we adjusted the position of the Leap Motion to make the users' hand visible.

Specifically, we place the Leap Motion sensor facing up at the right side of the laptop computer, or between the computer monitor and the keyboard of a desktop computer as shown in Figure 1.

We first collect normal data for 24 subjects independently, i.e., each subject writes the article without observing any other subjects. We use the first few paragraphs for enrollment and the rest for testing without cross validation. We get 3256 samples in total, and include 10 words in a sample on average. Among them, 720 samples are for training, and others are used for testing. Then, we collect data from 7 subjects who act as observing attackers. In total, 84 attack samples are collected.

6.1 Results on CR Authentication

6.1.1 Evaluation Metrics

In the experiment, we verify whether a testing sample is written by a given legitimate user or not. The output is binary – either 'yes' or 'no'. As discussed in Section 5.4, we train a binary SVM classifier for each legitimate subject and use samples from attackers and each legitimate user to test the performance of the system. Given the binary SVM output, we apply a threshold to determine if the testing sample belongs to this user. For evaluation, we compute the following metrics to examine the performance under attacks: (a) tp, the number of true positives, (b) tn, the number of true negatives, (c) fp, the number of false positives, and (d) fn, the number of false negatives.

Given that the training data samples can be randomly selected, we can conduct multiple rounds of experiments and calculate these four metrics at each round. This way, we can compute the true positive rate (TPR), and the false positive rate (FPR) as

$$TPR = \frac{\sum_{i=1}^{m} tp_i}{\sum_{i=1}^{m} tp_i + \sum_{i=1}^{m} fn_i},$$
$$FPR = \frac{\sum_{i=1}^{m} fp_i}{\sum_{i=1}^{m} fp_i + \sum_{i=1}^{m} tn_i}.$$

where $i = 1, 2, \cdots, m$ indicates the i-th round of experiments. We plot the standard ROC curves to quantitatively evaluate the performance of rejecting a skilled attacker. The **ROC curve** stands for the receiver operating characteristic curve and is a plot of the TPR against FPR by varying the threshold of the binary SVM classifiers. The closer the curve to the top-left corner $(0, 1)$, the better the authentication performance. An **Equal Error Rate (EER)** is the one where the FPR equals to the false negative rate (FNR). The smaller the EER, the better the performance.

6.1.2 Results for Non-Observing Attackers

Non-Observing Insiders. *Twenty-four classifiers are trained using the training data from 24 subjects and are tested using the remaining data. For each classifier, one subject act as the victim (positive training samples) and the rest 23 ones act as attackers (negative training samples). To test the classifiers, the data from the 24 subjects that are not used for training are used for testing.* To evaluate performance under Non-Observing

TABLE 2
The EER of one victim subject attacked by 23 Non-Observing insiders with various parameters. The results are averaged with 24 victim subjects. We varied one parameter at a time and use the following default parameters: one sample contains no more than 2,000 frames; each experiment uses 30 samples per subject for training and the rest for testing; each stroke segment contains 12 frames; each experiment has a vocabulary size of $K = 200$.

Sample Length (#Frames)	500	1,000	**2,000**	3,000	4,000	5,000
Words per Sample wps	2.5	5	**10**	15	20	25
Writing Time per Sample	4.4s	8.8s	**17.5s**	26.3s	35.1s	43.9s
Writing Time for Training (Total)	2.2min	4.4min	**8.7min**	13.5min	17.5min	21.9min
EER	6.79%	3.11%	1.18%	1.01%	0.38%	0.32%

No. of Training Samples	10	20	**30**	40	50
EER	2.35%	1.63%	1.18%	0.98%	0.91%

Stroke Segment Length (#Frames)	8	**12**	16	20
EER	2.48%	1.18%	1.34%	1.65%

Vocabulary Size K	50	100	**200**	400
EER	5.39%	3.54%	1.18%	0.87%

Style-level Feature	Histogram	Co-occurrence Matrix	Combined Both
EER	5.45%	1.18%	2.01%

insiders, we utilize the data from the 24 subjects and do not include data from subjects who act as observing attackers. Each subject is considered as a victim and the remaining 23 subjects are considered as attackers. Results are based on default parameters (one sample contains no more than $2,000$ frames; a training set contains 30 samples per subject, the rests are for testing; each stroke segment contains 12 frames; the vocabulary size equals to 200). Figure 11 shows the results where each ROC curve represents one subject. We observe that for almost all the subjects, the performance under Non-Observing attacks are close to ideal, near 100% TPR, 0 FPR. The worst cases are mainly caused by some of their samples that show similar styles (i.e., represented by similar co-occurrence matrices) to a few other subjects. We further calculated an average EER of 1.18% among the 24 subjects.

`Varying Parameters`. We conduct experiments with insider attacks to explore the effect of different parameters. In each experiment, we change one parameter, and keep other parameters unchanged. Table 2 summarizes the results when varying different parameters. The bold numbers are the default parameters we chose.

Length of the sample directly affects the usability of the system – the longer the sample, the longer time that takes to input a testing sample. As expected, with the increase of sample length L_s, the accuracy increases. To balance the usability and performance, we choose $L_s = 2,000$, since a sample length of $L_s = 3,000$ demands a user writes 10 more seconds to gain an decrease of 0.17% on error rate.

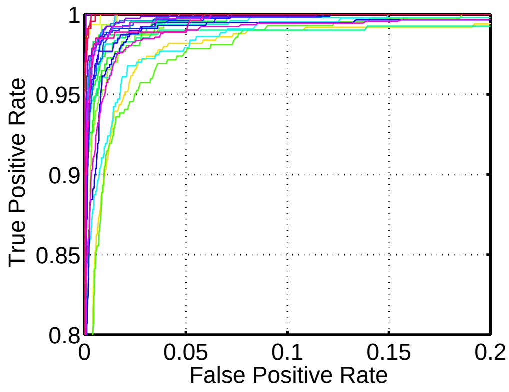


Fig. 11. Performance of all 24 subjects under Non-Observing attacks from insiders (samples from the rest of 23 subjects) presents by ROC curves. Each colored curve corresponds to the ROC of one subject.

The number of training samples. Our results show that a larger number of training samples leads to a better classification results. Conservatively, we choose 30 samples per subject for training, which maps to a training session less than 10 minutes, and has an error rate of 1.18%. For a system that can tolerate an error rate of 2.35%, 10 samples are enough for training, which requires a training input session of 3 minutes for each user. Note that the training input session could be reduced further if a higher error rate is acceptable.

Stroke segment length L_{ss}. We conduct experiments to find the length of stroke segments that can represent the coherent features of 3D human handwriting movements. The results in Table 2 show that $L_{ss} = 12$ maps to the best results among the experiments, thus we choose each stroke segment to be consists of 12 frames. Note that the overlapped ratio r is set as $2/3$ since experiment under this ratio performs better than other options.

The vocabulary size K indicates how many primitives are selected to represent the handwriting style. If the number of primitives is too small, then they cannot capture all the handwriting style. If the number of primitives is too large, then the size of co-occurrence matrix would be large and induce extra computing overhead. As expected, the results in Table 2 show that a larger K results in a better performance. Considering the overhead of feature calculation, we choose 200 as the size of the vocabulary.

Histogram vs. Co-occurrence Matrix. We conduct an experiment to justify choosing transition co-occurrence matrix of stroke segments as the style features. In this experiment, we directly use the histogram of the individual (quantized) stroke segment features (i.e., the histogram of the stroke segment indices), the EER is 6.45%, which is much higher than 1.18% that uses the proposed style features (i.e., co-occurrence matrix). In addition, we combine two features (both histogram and co-occurrence matrix), but EER (i.e., 2.01%) is not as high as the ones using only co-occurrence matrix. Therefore, we conclude that the stroke segment-transition statistics can better represent the handwriting style.

Non-Observing Impostors. *For each subject (one of the 24 subjects, s_i) that acts as a victim, we assume another subject (one from the rest of 23 subjects, s_j) is an impostor without knowing any information of the victim. We label the rest 22 subjects as s_{rest}. For each victim s_i, we train 23 classifiers and show average result as one ROC curve in Figure 12.* To train classifiers for s_i, we use s_i's training data as positive samples, the s_{rest}' training data as negative samples, while s_j's data are not included. In other word, the 23 classifiers for s_i have the same positive samples from s_i, but negative training samples are partly different as the s_{rest} sets are different since different s_j is chosen as an impostor. To test classifiers of s_i, we label all data from the attacker s_j as negative samples, and the testing data from the victim s_i are positive samples. The result of each curve shown in Figure 12 is the averaged testing results on the 23 classifiers.

In this scenario, the attackers' samples are not included in training. Among 24 subjects, we choose one of the subject as a victim and another subject as a non-observing impostor from the rest 23 subjects. The impostor's samples are never a part of the training sets except for building the vocabulary. All training data from the 23 subjects except the impostor one are the legitimate users for training. For each subject as a victim, we repeat the experiment 23 times by rotating the impostor role. Note that we call such an attacker as non-observing impostor because he/she is not part of the group of legitimate users and does not contain any useful information of the users. Results shown with ROC curves for all subjects are in Figure 12. The overall EER is 2.45% averaged for all 24 subjects.

6.1.3 Results for Observing Attacks

We use classifiers of 4 victims which are trained in Non-Observing Insiders' part and test the 4 classifiers with 7 attackers' data, whose data never be part of the training. We use observing attacks to emulate the extreme case where an attacker happens to observe a user' writing process and the system ask the attacker to respond to the same challenge as the victim have met, although it is unlikely for the system to select the same challenge in practice. In this experiment, out of the 24 subjects, we randomly select 4 as victims, and invite 7 subjects act as attackers. Against each victim, each attacker writes a selected paragraph of the article three times. Before each new attempt, the attackers spend time to 1) observe the victims' writing processes by shoulder surfing, and 2) view the finger tracking results of the victims' handwriting on the computer screen. This way, we collected $4 \times 7 \times 3 = 84$ writings as the observing attack data.

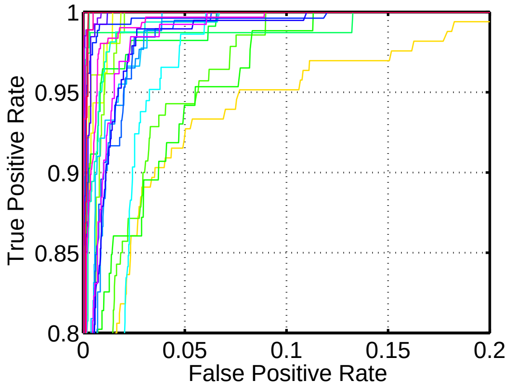


Fig. 12. Performance of all 24 subjects rejecting impostors presents by ROC curves. Each colored curve corresponds to the ROC of one subject.

In the observing attacks, we use the same binary classifiers of the 4 victims trained in the Non-Observing attack scenarios. This way, none of the observing attack data are included for training. We then use the victims' untrained normal data and the collected observing attack data for evaluation. Figure 13 shows the verification performance of these 4 victim subjects against the observing attacks. Our results show that the observing attackers cannot achieve a better verification result than a random person who has no information about the victim. In particular, the averaged EER of Non-Observing insider attacks is 1.16% for the 4 victims, which is similar to the average on of 24 subjects under insider attacks. The averaged EER of observing attacks is 3.11% for these 4 victims. From the results, we observe that the learning the writing content does not necessary always improve the impersonate attacks and may have limited help in impersonating users in MoCRA.

7 Discussion and Conclusion

7.1 Discussion

Content Matching. In this paper we focused on how to apply the in-air handwriting style for authentication. For security concerns, we introduced challenge-response procedure. We believe that the matching between the challenge and the response (i.e., handwriting content matching) can utilize approaches designed for handwriting recognition. Much work has been proposed to recognize content from off-line writings, on-line writings (2D), and in-air writings (3D). These content matching approaches can be added to fulfill the proposed method. Essentially, the implemented MoCRA in the paper authenticates based on 'how a user writes' instead of both 'how a user writes' and 'what a user writes'. As a direction for future work, it is important to investigate content matching and its impact on the accuracy of the system. We suspect that content matching could improve the security, since our evaluation results do not reject the attack

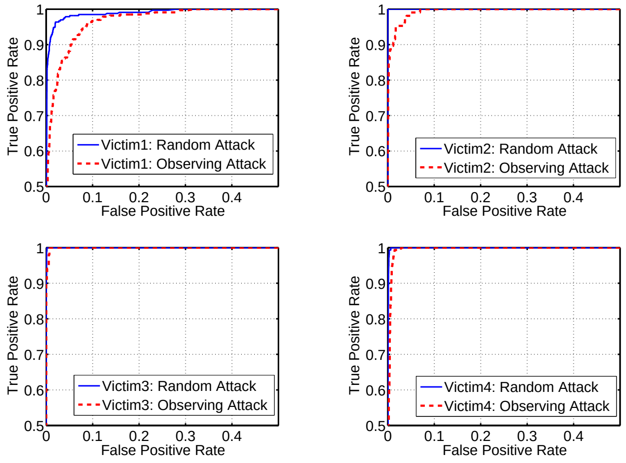


Fig. 13. Authetication performance of the victims under observing attacks presents by ROC curves.

samples that have the same hand writing style distribution as the legitimate one but extracted from different content, such cases should be rejected at the earlier stage – content matching. In addition, as a direction of future work, it is worth performing usability studies to understand the trade-off between security benefits and extra effort imposed by the introducing of the challenge-response mechanism.

Handwriting Segmentation. We used a fix number of continuous sampling frames to represent the primitives of handwriting. However, based on the trajectories' inner variations, one can extract segments according to a few features (e.g., curvature, direction, speed) so that each segment will contain unique style information of a user, while the length of each segment is not necessarily the same. Thus, it is worth exploring new segmentation methods for enhancing writing style modeling as a task of future work.

Vocabulary Generation. We extracted a vocabulary based on the training data prior to classifier training. Ideally, pre-trained vocabulary with a separate database can avoid new vocabulary training whenever a new user enrolls. For scenarios of a large group of users, the pre-training on the vocabulary in a separate database can improve the computation performance at the training stage. However, our experiments involved 24 subjects on campus, and is not large enough to generate a representative vocabulary for various writing styles while maintaining a statistically significant results of authentication evaluation. We note that this limitation can be eliminated by extra data collection.

7.2 Conclusion

We design a motion-based challenge response authentication scheme that is based on a user's handwriting style in a 3D space, and we leverage a hand motion sensor — Leap Motion controllers — to capture finger movements as a user writes in the air. Our scheme authenticates users based on 'what they write' and 'how they write'. We focus on 'how they write' in this paper, and 'what they write' will be discussed in the future work. Our results show that the co-occurrence matrix that built on sets of stroke segments (i.e., a small, fixed-length trajectory of fingertip movements) can model a user's handwriting style. We envision that our authentication scheme can be used in applications such as building security guard authentication.

We built a system called the MoCRA and evaluated it on 24 subjects for 7 months. Our results show that MoCRA can reliably authenticate one of the 24 subjects with an average equal error rate of 1.18% and reject impostors with an error rate of 2.45%. In addition, MoCRA can effectively reject skilled attackers that observed victims' writing process multiple times.

Acknowledgment

This work was supported in part by UES Inc./AFRL-S-901-486-002, NSF-1658987, NSFC61672376 and NCPTT-P16AP00373. The authors also would like to thank all volunteers for their help collecting data and Carter Bays for improving the paper.

References

[1] S. Uellenbeck, M. Durmuth, C. Wolf, and T. Holz, "Quantifying the security of graphical passwords: The case of android unlock patterns," in *Proceedings of ACM CCS*, 2013, pp. 161–172.

[2] F. Monrose, M. K. Reiter, and S. Wetzel, "Password hardening based on keystroke dynamics," in *Proceedings of the 6th ACM conference on Computer and communications ecurity*, ser. CCS '99, 1999, pp. 73–82.

[3] 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*, ser. CCS '11, 2011, pp. 139–150.

[4] A. A. E. Ahmed and I. Traore, "A new biometric technology based on mouse dynamics," *IEEE Transaction on Dependable and Security Computing*, vol. 4, no. 3, pp. 165–179, 2007.

[5] A. Juels and R. R. (MIT), "Honeywords: Making password-cracking detectable," in *Proceedings of ACM CCS*, 2013, pp. 145–160.

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

[7] M. Gorman, "Leap motion controller review," http://www.engadget.com/2013/07/22/leap-motion-controller-review/.

[8] L. Schomaker, *Writer Identification and Verification*. Springer London, 2008.

[9] C. C. Tappert, C. Y. Suen, and T. Wakahara, "The state of the art in online handwriting recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 8, pp. 787–808, Aug. 1990.

[10] S. Vikram, L. Li, and S. Russell, "Handwriting and gestures in the air, recognizing on the fly," in *In Proceedings of the CHI 2013 Extended Abstracts*, Paris, France, 2013.

[11] R. A. . S. S. . A. M. N. . J. S. . C. V. Jawahar, "Online handwriting recognition using depth sensors," in *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, Aug 2015.

[12] L. Motion, "Leap overview," https://developer.leapmotion.com/documentation/Languages/CSharpandUnity/Guides/Leap_Overview.html.

[13] "How does the leap motion controller work?" http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work.

[14] D. DAVIES and W. PRICE, *Security for computer networks*. Chichester, England: John Wiley, 1984.

[15] S. M. Bellovin and M. Merritt, "Encrypted key exchange: Password-based protocols secure against dictionary attacks," in *IEEE SYMPOSIUM ON RESEARCH IN SECURITY AND PRIVACY*, 1992, pp. 72–84.

[16] L. OGorman, "Comparing passwords, tokens, and biometrics for user authentication," in *Proceedings of the IEEE*, vol. 91, no. 12. IEEE, 2003, pp. 2019–2040.

[17] R. Johnson, W. J. Scheirer, and T. E. Boult, "Secure voice based authentication for mobile devices: Vaulted voice verification," *Proceedings of SPIE, Biometric and Surveillance Tech. for Human and Activity Identification*, vol. 8712, May 2013.

[18] K. Revett, "A bioinformatics based approach to user authentication via keystroke dynamics," *International Journal of Control, Automation and Systems*, vol. 7, no. 1, pp. 7–15, 2009.

[19] Z. Jorgensen and T. Yu, "On mouse dynamics as a behavioral biometric for authentication," in *Proceedings of the ASIACCS2011*, 2011.

[20] D. Gafurov, E. Snekkenes, and P. Bours, "Gait authentication and identification using wearable accelerometer sensor," in *Automatic Identification Advanced Technologies, 2007 IEEE Workshop on*, 2007.

[21] M. Derawi and P. Bours, "Gait and activity recognition using commercial phones," *Computers and Security*, vol. 39, pp. 137–144, Nov. 2013.

[22] N. Sae-Bae, K. Ahmed, K. Isbiste, 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*, ser. CHI '12, 2012, pp. 977–986.

[23] M. Sherman, G. Clark, Y. Yang, S. Sugrim, A. Modig, J. Lindqvist, A. Oulasvirta, and T. Roos, "User-generated free-form gestures for authentication: Security and memorability," in *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '14, 2014, pp. 176–189.

[24] E. R. M. Gabel, R. Gilad-Bachrach and A. Schuster, "Full body gait analysis with kinect," in *Proceedings of EMBC 2012*. Annual

International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), August 2012.

[25] E. Hayashi, M. Maas, and J. I. Hong, "Wave to me: User identification using body lengths and natural gestures," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '14. New York, NY, USA: ACM, 2014, pp. 3453–3462.

[26] I. Aslan, A. Uhl, A. Meschtscherjakov, and M. Tscheligi, "Mid-air authentication gestures: An exploration of authentication based on palm and finger motions," in *Proceedings of the 16th International Conference on Multimodal Interaction*, ser. ICMI '14. New York, NY, USA: ACM, 2014, pp. 311–318.

[27] J. Tian, C. Qu, W. Xu, and S. Wang, "Kinwrite: Handwriting-based authentication using kinect," in *NDSS*, 2013.

[28] I. Nigam, M. Vatsa, and R. Singh, "Leap signature recognition using hoof and hot features," in *Image Processing (ICIP), 2014 IEEE International Conference on*, Oct 2014.

[29] M. Bulacu, L. Schomaker, and L. Vuurpijl, "Writer identification using edge-based directional features," in *Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 2*, ser. ICDAR '03, Washington, DC, USA, 2003.

[30] D. S. Guru and H. N. Prakash, "Online signature verification and recognition: An approach based on symbolic representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, pp. 1059–1073, 2009.

[31] M. Bulacu and L. Schomaker, *Writer Style from Oriented Edge Fragments*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 460–469.

[32] S. He and L. Schomaker, "Delta-n Hinge: Rotation-Invariant Features for Writer Identification," in *Proceedings of the 22nd International Conference on Pattern Recognition*. IEEE, 2014, pp. 2023–2028.

[33] F.-F. Li and P. Perona, "A bayesian hierarchical model for learning natural scene categories," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02*, ser. CVPR '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 524–531.

[34] M. Bulacu, L. Schomaker, and A. Brink, "Text-independent writer identification and verification using textural and allographic features," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 29, p. 2007, 2007.

[35] L. Schomaker and M. Bulacu, "Automatic writer identification using connected-component contours and edge-based features of uppercase western script," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 6, pp. 787–798, 2004.

[36] M. Liwicki, A. Schlapbach, H. Bunke, S. Bengio, J. Mariéthoz, and J. Richiardi, *Writer Identification for Smart Meeting Room Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 186–195.

[37] B. Li, Z. Sun, and T. Tan, *Online Text-Independent Writer Identification Based on Stroke's Probability Distribution Function*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 201–210.

[38] B. Li and T. Tan, "Online text-independent writer identification based on temporal sequence and shape codes," in *ICDAR*. IEEE Computer Society, 2009, pp. 931–935.

[39] C. C. Tappert, C. Y. Suen, and T. Wakahara, "The state of the art in online handwriting recognitionr," in *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 12, August 1990, pp. 787–808.

[40] H. Lee and B. Verma, "Binary segmentation algorithm for english cursive handwriting recognition." *Pattern Recognition*, vol. 45, no. 4, pp. 1306–1317, 2012.

[41] S. Madhvanath and A. Bharath, "Hmm-based lexicon-driven and lexicon-free word recognition for online handwritten indic scripts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 670–682, 2012.

[42] R. Plamondon and S. N. Srihari, "On-line and off-line handwriting recognition: A comprehensive survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 1, pp. 63–84, Jan. 2000.

[43] G. Dimauro, S. Impedovo, and G. Pirlo, "Component-oriented algorithms for signature verification," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 8, no. 3, pp. 771–794.

[44] J. Park, V. Govindaraju, and S. N. Srihari, "Efficient word segmentation driven by unconstrained handwritten phrase recognition," in *Document Analysis and Recognition, 1999. ICDAR '99. Proceedings of the Fifth International Conference on*, 1999, pp. 605 – 608.

[45] S. H. L. Schomaker, "A polar stroke descriptor for classification of historical documents," in *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, Aug 2015.

[46] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Prentice Hall, 2008.

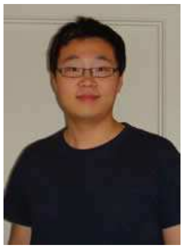
[47] T. Joachims, "Making large-scale svm learning practical. advances in kernel methods-support vector learning," 1999.

[48] C.-C. Chang and C.-J. Lin, "Libsvm – a library for support vector machines," https://www.csie.ntu.edu.tw/~cjlin/libsvm/.

[49] J. D. Hincapié-Ramos, X. Guo, P. Moghadasian, and P. Irani, "Consumed endurance: A metric to quantify arm fatigue of mid-air interactions," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '14. New York, NY, USA: ACM, 2014, pp. 1063–1072.

[50] A. Cockburn, P. Quinn, C. Gutwin, and J. Looser, "Air pointing: Design and evaluation of spatial target acquisition with and without visual feedback," *International Journal of Human-Computer Studies*, vol. 69(6), pp. 401–414, 2011.

Jing Tian is a Ph.D. student in University of South Carolina in Computer Science and Engineering. She received her B.S. degree in Automation from Northeastern University, China in 2006, an M.S. degree in Navigation, Guidance and Control from Northeastern University, China in 2008. Her research interests include user authentication, biometrics, machine learning, and gesture recognition.



Yu Cao is currently a software engineer at Facebook. He received a B.S. degree in Information and Computing Science from Northeastern University, China in 2003, an M.S. degree in Applied Mathematics from Northeastern University, China in 2007, and a Ph.D. degree in Computer Science and Engineering at the University of South Carolina in 2013. His research interests lie in computer vision, medical image processing, machine learning and pattern recognition.

Wenyuan Xu is currently a professor in the College of Electrical Engineering at Zhejiang University. She received her B.S. degree in Electrical Engineering from Zhejiang University in 1998, an M.S. degree in Computer Science and Engineering from Zhejiang University in 2001, and the Ph.D. degree in Electrical and Computer Engineering from Rutgers University in 2007. Her research interests include wireless networking, smart systems security, and IoT security. Dr. Xu received the NSF Career Award in 2009 and was selected as a young professional of the thousand talents plan in China in 2012. She was granted tenure (an associate professor) in the Department of Computer Science and Engineering at the University of South Carolina in the U.S. She has served on the technical program committees for several IEEE/ACM conferences on wireless networking and security. She has published over 60 papers and her papers have been cited over 3000 times (Google Scholar).

Song Wang received the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana-Champaign (UIUC), Champaign, IL, USA, in 2002. From 1998 to 2002, he worked as a Research Assistant with the Image Formation and Processing Group, Beckman Institute, UIUC. In 2002, he joined the Department of Computer Science and Engineering, University of South Carolina, Columbia, SC, USA, where he is currently a Professor. His research interests include computer vision, image processing, and machine learning. Prof. Wang is a Senior Member of the IEEE Computer Society. He is currently serving as the Publicity/Web Portal Chair of the Technical Committee of Pattern Analysis and Machine Intelligence of the IEEE Computer Society and an Associate Editor of Pattern Recognition Letters.