Motion-Based Challenge-Response Authentication and Continuous Identification using Depth Sensor

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 obtained the secret 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. Utilizing Leap Motion, MoCRA captures the user's writing movements and then extracts his 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 correctly 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 imposers with 2.26% 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 [2], [24], [33], [49], [52]. 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, essentially, such a challenge response method authenticates based on *what* you know instead of *who* you are. Anyone knowing the shared secret can pass authentication. Such authentication cannot prevent insider attacks, 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 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.

Both physiological and behavior biometrics can be used to identify who you are. Considering that physiological biometrics are not privacy-friendly, we focus on designing a behavior-biometric-based CR authentication and user identification, which called MoCRA. We utilize data extracted from finger movements when a user writes in the air, which are motion data with limited privacy concerns. 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 maps to the user. Since handwriting recognition can utilize existing technology [46] and recent study shows that Leap Motion has the potential for handwriting recognition application [50], this paper focuses on the second step — user verification based on the handwriting.

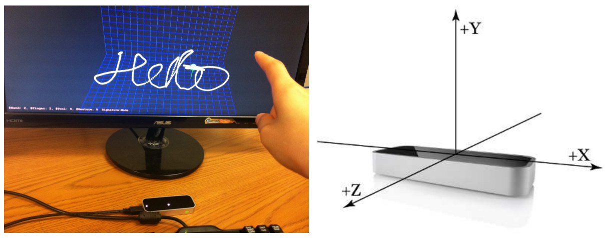

Fig. 1. An illustration of using a Leap Motion controller to acquire a user's handwriting in the 3D space for content independent re-authentication.

2016 Besides one time authentication, continuous identification are neccesary in some scenarios such as smart meeting [], which aims at automate meeting tasks such as note tracking and extracting important issues, etc. We envision in the future we may have a mid-air writing input device such as advanced leap motion. After initial user authentication, attendees will enter their comments by writing towards a

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

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

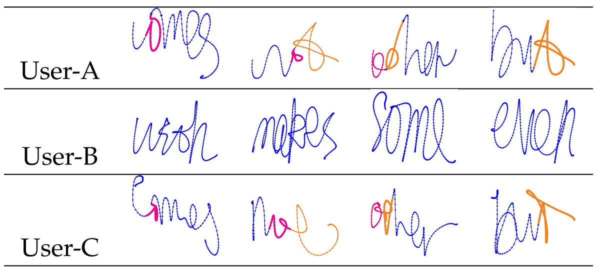

Fig. 2. Examples of processed trajectories written by three users. Despite that some handwritten words by user-A and user-C appear to be similar (e.g., `but`) yet some letters written by the same user may differ (e.g., the highlighed letters 'o' and 't'), our re-authentication system can correctly distinguish them regardless of what they wrote. This is because `MoCRA` is independent of text contents and utilizes features derived at the scale of a writing component, e.g., a small segment composing trajectories.

leap motion, since writing math symbols or formulas is preferable over typing. Throughout the meeting, `MoCRA` can harvest behavioral biometrics that is embedded in writing regardless of the writing content, so that it can verify known users (i.e., identify a legitimate user out of a set of known candidates) and raise an alarm when an alien is detected (i.e., an attacker tries to impersonate known candidates).

To track the motion of human fingers, we utilize depth sensors, because they are capable of capturing 3D human motion in a *touch free* manner, just like how a camera captures pictures. Unlike behavior biometrics that rely on *contact-based* input devices, such as touch screens [27], [29], mouse movements [2], [52], keystroke dynamics [33], [40], depth sensors eliminate hygiene concerns and smudge attacks [4]. Without loss of generality, we focus on utilizing an emerging popular depth sensor — a Leap Motion controller (in short, Leap Motion), which is designed to let users interact with a computer by waving their hands and fingers [16], as shown in Figure 1. Leap Motion captures the 3D motion of fingers at a relatively high precision and tracks the finger length as well as width. This allows capturing behavior biometrics at a higher degree of freedom (i.e., entropy) than the traditional handwriting captured on flat surfaces (e.g., paper).

2016 `MoCRA` has to be *content-independent* for both CR authentication and continuous identification. Because the CR authentication can require a user to write any content including letters, digits, and special symbols; because during the meeting the attendees can write any format of notes. We design `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 is non-trivial because `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). In addition, `MoCRA` has to reject attackers. To characterize handwriting styles, we introduce the concept of writing components: a component is a small segment of finger trajectories and is small enough to serve as building blocks for representing a handwriting style. To make the classification computationally reasonable, we propose a transition co-occurrence matrix for feature reduction. Combining a co-occurrence matrix with an effective classification method — Support Vector Machine (SVM) — we are able to re-authenticate users correctly. For example, Figure 2 illustrates handwriting of three users. Despite that some handwritten words by user-A and user-C 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 correctly distinguish them regardless of what they wrote. This is because `MoCRA` is independent of text contents and utilizes features derived at the scale of a writing component. The main contributions of this paper are listed below.

- We designed a motion-based challenge-response authentication and continuous identification that we call `MoCRA`, which 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 along with a co-occurrence matrix through which we can reduce the feature dimension 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. Our system can not only continuously identify legitimate users but also reject impostors.
- We evaluated `MoCRA` on 24 subjects over 7 months, including 7 observing attackers and an emulated motion tracking robot arm. The CR authentication results show that the average error rate is 1.18% for random insider attacks and 2.26% for random impostors. The identification results show that using 8.8 seconds of handwriting samples, the average identification accuracy is 93.4%, and using 17.5 seconds of handwriting samples, and average accuracy is 97.4%. In addition, `MoCRA` can reliably reject observing attackers and a robot arm that synthesizes writings.

2 Background and Overview

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 identity authentication.

In this section, we define attack models, provide background on Leap Motion, discuss the basic idea of modeling handwriting styles, and overview our authentication system.

2.1 Attack Model

We assume that the system is used in a well controlled environment and attackers have no physical access to hardware (e.g., Leap Motion) nor software (e.g., operating systems, database). Second, the communication path between 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 Leap Motion while mimicking other users. we consider the following three types of attack models.

1) **Random 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, Random 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.

3) **Naive robot arm attackers** are introduced to emulate a super human who can precisely reproduce the recorded finger movements, so that we can evaluate the upper bound of the impact of shoulder surfing. Since `MoCRA` adapts CR mechanism, the naive robot arm has to write different content to take the challenge. We assume that a naive robot-arm can record some observations of the victim and split writing trajectories that correspond to each letter, and synthesize the writing of the new word by sequentially linking the writing trajectory of each letter of the required string. We assume the robot arm does not have unlimited access to the victim, thus it is not capable of get transitions between letters, which demands the recording of 26! types of transitions.

2.2 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 Microsoft Kinect, Leap Motion tracks the hands and fingers (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 [1]. 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), the trajectory of a 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) [1], [35]. 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.

Since its release in July 2013, Leap Motion has hundreds of applications on its app store and was used to operate computers by 3D hand motion. It is also an option for free hand TV control application [19]. In summary, Leap Motion is gaining its popularity and is ideal for capturing handwriting motion.

2.3 Characterize Handwriting Styles

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 can rely on existing technologies [47], and in this section we discuss 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 content) of the same user and occasional handwriting similarity (especially in writing the same content) between different users. The key to overcome the challenges is to characterize handwriting styles effectively and efficiently.

Writing Components 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

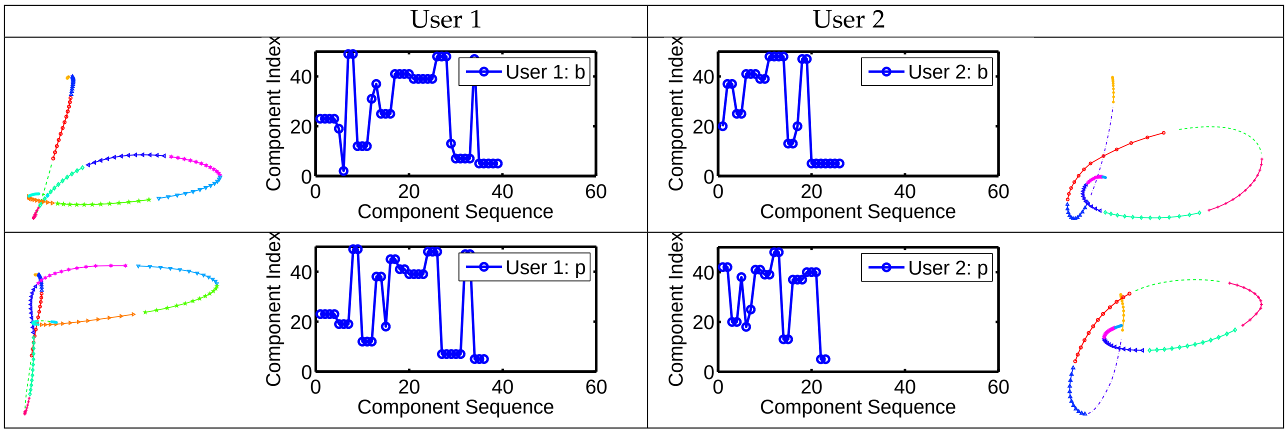

Fig. 3. An illustration that characters written by the same user exhibit similarity while the ones by different users exhibit difference. The motion trajectories are divided into components, and the plots show the component-index (y-axis) change over time (x-axis). Plots with similar profiles represent the similarities of the handwriting style.

precisely, as shown in Figure 2. Although content recognition is possible, perfectly delimiting the fingertip trajectory of each letter is challenging. Thus, instead of characterizing handwriting style by each individual symbols, we model it To avoid the burden of extracting individual symbols, we choose to characterize handwriting styles by short-length continuous trajectories, which are analogous to strokes [12], [38]. 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 *writing components*. To reduce the impact of the starting point on a trajectory for component partition, we apply a temporal-sliding window over the fingertip trajectory for constructing components, and the constructed components can be partially overlapped.

Writing components 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 writing components. Thus, the two fingertip trajectories created by the same user when writing different sets of words could contain a large percentage of the similar writing components. The writing components belonging to different users typically show little similarity. For instance, as shown in Figure 3, the letters 'b' and 'p' have similar composition. Some components (denoted by different colors on the characters) of 'b' and 'p' from the same user (either User-A or User-B) show similarity, but the components of User-A are different from the ones of User-B. Thus, it is imaginable that the trajectories of the word `'bob'` and `'pop'` from the same user may have many similar components, but the ones from different users may share few similar components.

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 writing components, we can concatenate the frame-level features of all frames of a component to form *component-level features*. Then, we can combine all the component-level features of a trajectory sample to construct *sample-level features*, which represent the handwriting style. However, simply combining all the component-level features will lead to high-dimensional vectors. To address this issue, we can define a component vocabulary that best represents the collection of components of all the enrolled users. The vocabulary consists of a smaller set of primitives corresponding to typical components of those users, and each primitive will be assigned a unique index. The creation of a component vocabulary can be conducted during the training phase. During the testing stage, each component is assigned the index of its nearest primitive. This way we reduce the high-dimensional component-level features to an index.

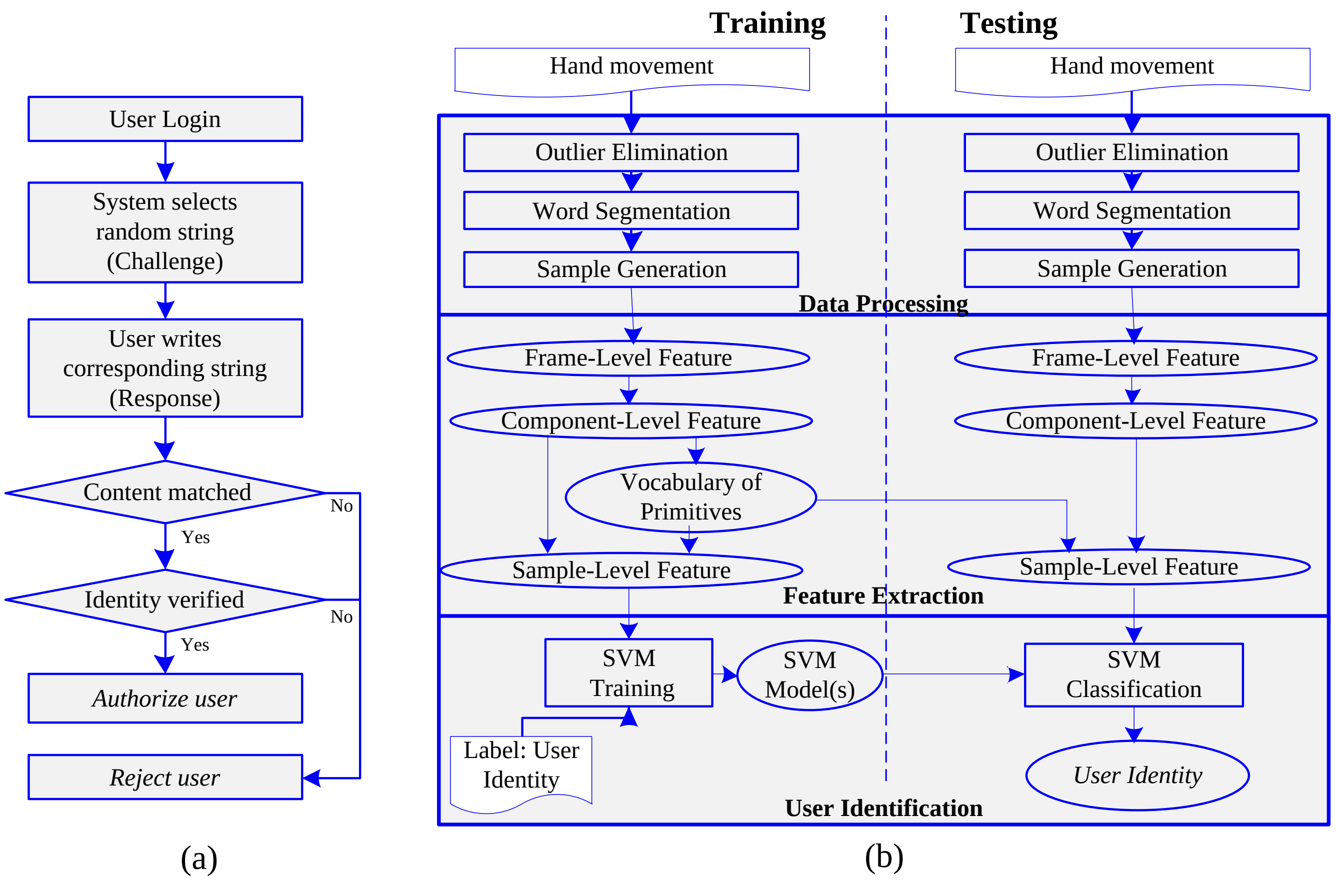

Fig. 4. (a)Flow chart of `MoCRA` system. 2016(b)Flow chart of identity matching (also as the flow chart of continuous identification). Details of the 3-Level feature extraction are shown in Figure 7.

Finally, to compare the similarity between handwriting styles of multiple samples, we have to perform statistical analysis to achieve content-independence. Thus, we obtain statistics on the component indices within a trajectory sample for constructing a low-dimensional feature for a sample. Instead of using the histogram of the individual component indices (widely used in the literature), we examine the temporal transition between components. The intuition is that a user may tend to write the same sequence of components, and such a sequence may be essential to represent handwriting styles. Specifically, we construct a co-occurrence matrix that counts the number of occurrences of each possible component (index) transition between temporally adjacent component pairs. This co-occurrence matrix reflects the distribution of the temporal component transition and we reshape it into a vector as a feature vector of a sample.

Continue with the example illustrated in Figure 3, 50 primitives were derived to form a vocabulary for illustration. We observe that the similar letter sets written by the same user contain similar component indices and transition pattern of component indices, while the component sequences of the same letter written by different users share few similarities. In particular, we notice that the index sequence of character ’b’ and ‘p’ written by the same subject show strong similarity, while the same characters written by different users exhibit difference. This example encourages us to study the effectiveness of using vocabulary and transition between components to model the handwriting style.

2.4 MoCRA System Overview

The MoCRA system consists of a Leap Motion for capturing fingertip movements in the 3D space, and a computing unit for processing challenge-response, which utilizes a supervised machine learning technique for identity authentication.

We design MoCRA system which includes a select of a random strings from a dictionary. Then Figure 4(a) shows a the flow chart of the challenge-response process. When a new user login, he/she have to input his rej

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 the MoCRA systems.

Content Matching. The focus of this paper is to study the biometric built on handwriting motion instead of content check, because several literature can achieve the content check. For instance, MoCRA can utilize online handwriting recognition that has been studied in pattern recognition community since 1980’s, 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 [28], [32], [39], [47], 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.

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, component-level features, and sample-level features. Level by level, the MoCRA system is able to derive the features that effectively and efficiently model the handwriting styles of users.

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

3 Data Processing

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

3.1 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*.

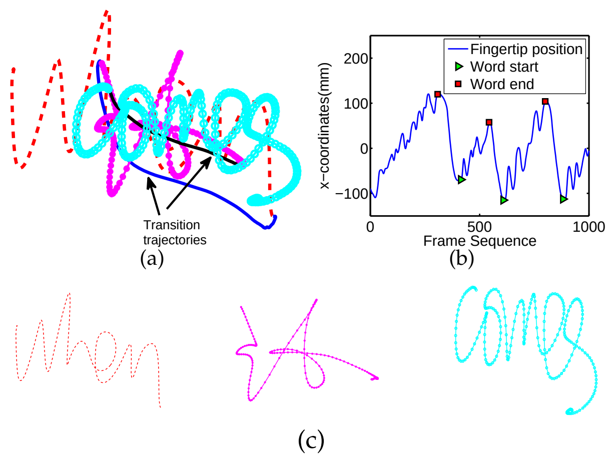

Fig. 5. An illustration of word segmentation. (a) what was recorded by the Leap Motion controller, which contains three words and transition trajectories connecting them. (b) word segmentation results on x-axis. The trajectories starting from a red square and ending at the next green square are the transition ones, which are removed to obtain word segments. (c) shows the separated words from the trajectory in (a).

Sample	s1				s2	
# Frames	231	169	231	149	260	144
User 1						

Sample	s1		s2		s3	
# Frames	466	288	430	183	512	264
User 2						

Fig. 6. 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 each sample length L_s is no larger than 800 frames, 6 words form different numbers of samples for Users 1 and 2.

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.

3.2 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 frames. 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 sequences. We can perform the remaining data processing and feature extraction steps on each sequence independently. For unlikely-position cases, we apply a temporal median filter to remove the outliers [14]. At each frame, we examine a temporal window centered at this frame, search for the median of all the fingertip positions in this window, and set the median as the new fingertip position of this frame. The median filter is performed on the x, y and z axes independently. For both cases, our algorithm only removes the outliers and preserve the smoothness and continuity of the handwriting trajectory.

3.3 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 the left to the right without any text overwriting. However, Leap Motion has limited range 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 end. 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 5 (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 the left to the right, the x-coordinates increases gradually. In comparison, quickly 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 5(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 (starting from a red square and ending at the next green square as shown in 5(b)), we obtain a set of disjoint word segments shown in 5(c)).

3.4 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 assure 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 word segments. Figure 6 demonstrates an example. In this example, we construct samples with a sample length close to 800 frames: we concatenate as many word segments as possible into a sample until one extra segment will make the sample contain more than 800 frames. With 6 word segments, user 1 can only create less than 2 samples, and user 2 can create almost 3 samples.

4 Feature Extraction and Classification

In this section, we elaborate how to extract features at the frame-, component- and sample-levels, respectively (shown in Figure 7). Given a sample, we extract frame-level features (denoted by f_k^i for the i-th frame) and then combine the features of L_c consecutive frames to generate a component-level feature (denoted by f_c^i for the i-th component). After finding the index of the primitive that is closest to each component, we obtain a sequence of indices. Then, the occurrences of component-transition pairs are calculated for creating a sample level feature. Then, we discuss the SVM classification for verifying users.

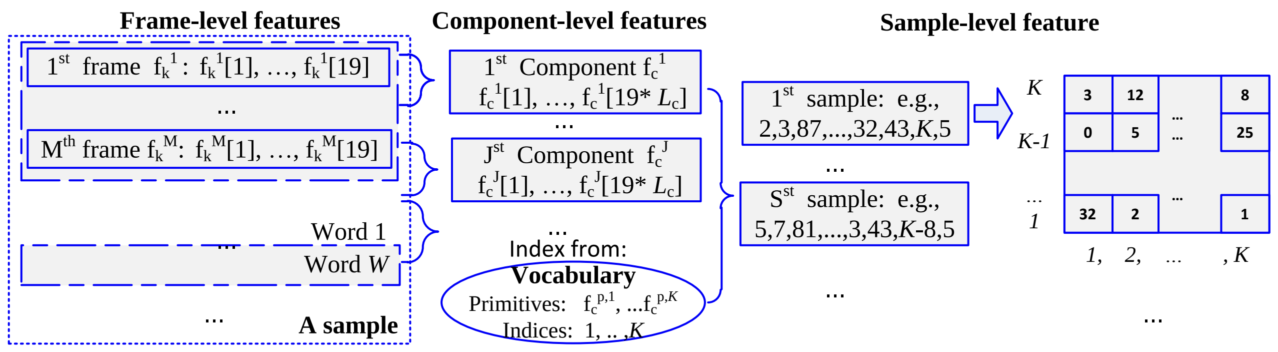

Fig. 7. An illustration of constructing a sample-level feature, i.e., co-occurrence matrix.

4.1 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 components comparable, all the fingertip positions within a writing component are normalized by subtracting the fingertip position in the first frame of the writing component. This way, in each component, the fingertip position in the first frame 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 $\mathbf{D}(t) = (D_x(t), D_y(t), D_z(t))^T$.

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 8.

9) **Curvature**. As show in Figure 8, 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 each dimension of these features such that it conforms to a standard Gaussian distribution in each word.

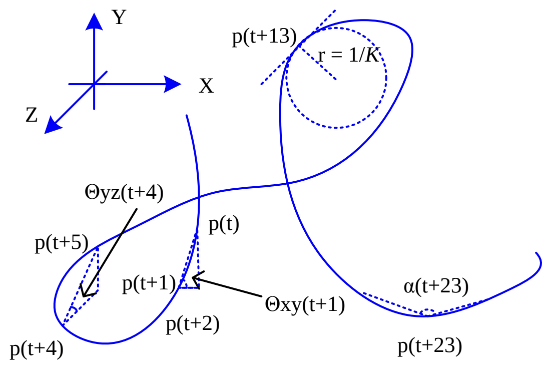

Fig. 8. An illustration of slope angle, path angle and curvature.

4.2 Component-Level Feature

As mentioned above, we construct short, partially overlapped, and fixed-length writing components by dividing the word segments of a sample. Let the length of a writing component be L_c, 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 components divided by the component length L_c. Then the i-th writing component starts at the frame of $(1-r)L_c * i$ and its length is L_c. This way, we can construct around $\left\lfloor\frac{L_w}{(1-r)L_c}\right\rfloor$ components 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 $\sum_{i=1}^{n}\left\lfloor\frac{L_w^i}{(1-r)L_c}\right\rfloor$ components.

To combine the frame-level features in a component into a component-level feature, we sequentially concatenate the features extracted from all the frames. With 19-dimension feature at frame level, the dimension of this component-level feature is $19 * L_c$.

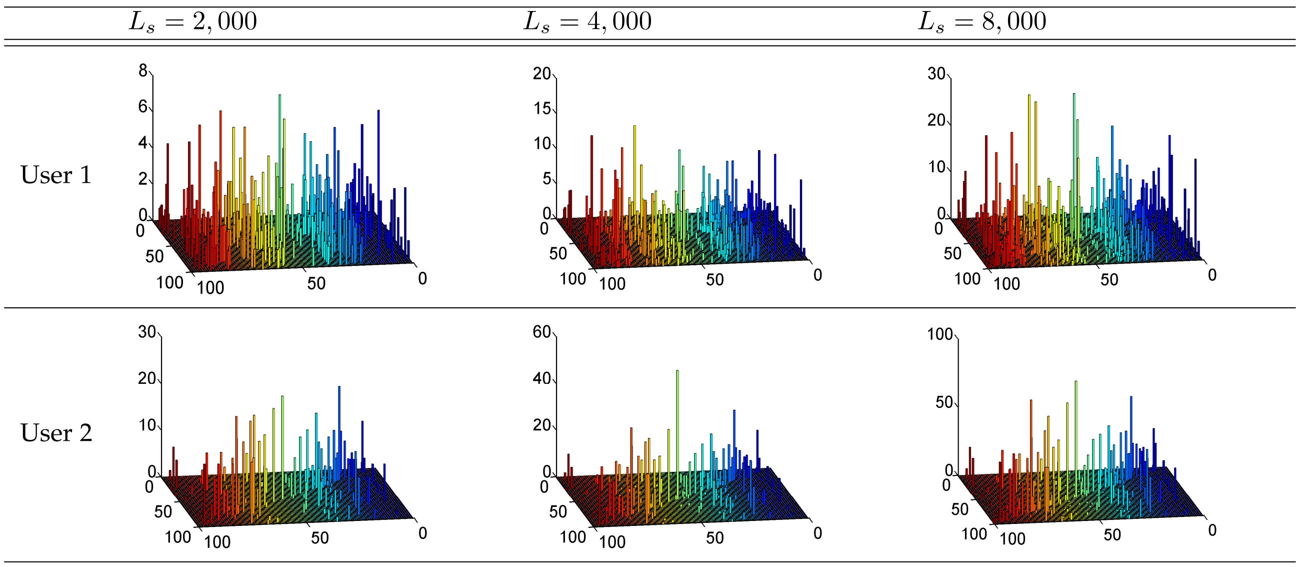

Fig. 9. Sample-level features (co-occurrence matrix) extracted from samples of two users. We choose $K = 100$ for illustration purpose. As the sample length increases, the shape of co-occurrence matrices remain similar, but the absolute value for each element (i.e. the occurences) increases. The results suggest that the matrices can model handwriting styles.

4.3 Sample-Level Feature

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

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

To reduce the dimension of sample-level features, we first quantize the component-level features. We use training samples to learn component primitives. Specifically, for each sample, we extract all the components from all the training samples. We then use K-means algorithm to group the component-level features into K clusters. The center of each cluster is considered as a primitive of component-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 component-level feature, for both the training samples and the testing samples, by finding its nearest primitive and assigning the component with the index of the corresponding primitive.

We then construct the sample-level feature by examining the transition of consecutive components. Specifically, for each pair of sequential components (with an offset of $(1-r)L_c$ 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 components. Scanning all such component pairs in a sample, we can build a $K \times K$ co-occurrence matrix, in which the ij-th element indicates the number of $< i, j >$ component transitions in this sample. We finally reshape this matrix into a K^2 dimension vector as the feature of this sample. Figure 9 shows examples of the constructed sample-level features in the form of the co-occurrence matrix, from two users' samples. As the sample length increases, the absolute value for each element (i.e., the occurrences) increases, but the shape of co-occurrence matrices remain similar. This suggest that the matrices can model handwriting styles.

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	$\log \frac{1}{\kappa(t)}$

4.4 Classification

We choose SVM [21] as the classification algorithm, because SVM is relatively efficient and showed accurate classification results in many real-life systems. 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. For each user, we take training samples from this users as positive samples and the other users' training samples as negative samples. Given a new test sample and the identity as a user name it claims, we apply the binary SVM classifier of this user. The sample will be authorized if the classifier returns positive. In addition, the classifiers should reject any impostors, the samples of which are never part of the training data. 2016 For continuously identification, we train a single classifier for all the users using multiclass-SVM based on the data collected at the beginning. Then we sample user's handwriting data required frequency. Given a new sampled test sample, we apply it to binary SVM classifiers and choose the identity of the binary classifier that returns the highest classification score.

5 Experiment and Evaluation

We evaluate the performance based on the data collected from 24 subjects, who are mainly female and male 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 in front of Leap Motion from time to time over a period of 7 months. The word length varies from 1 to 14 characters. In our first experiment, we use the first few paragraphs (about 2.2 - 8.8 minutes of handwriting) for enrollment and the rest for testing.

Writing in the air is prone to cause arm fatigue [18]. Since the least physically demanding position keeps the upper-arm at rest [9], 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 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. Then, we collect data from observing attackers.

5.1 Results on CR Authentication

5.1.1 Evaluation Metrics

In the experiment, we verify whether a testing sample is written by one of the legitimate users or not. The output is binary – either 'yes' or 'no'. As discussed in Section 4.4, we train a binary SVM classifier for each legitimate subject and use samples from attackers 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 lower the EER, the better the performance.

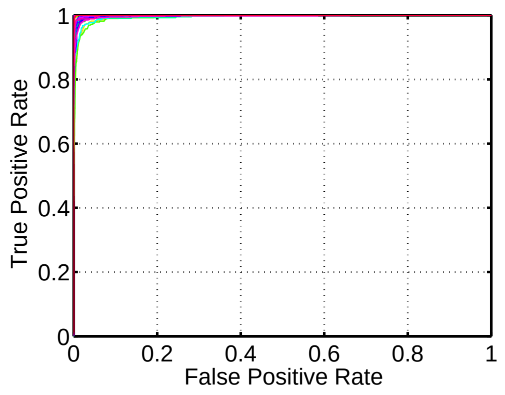

Fig. 10. Performance of all 24 subjects under random attacks from insiders presents by ROC curves. Each colored curve coresponds to the ROC of one subjct.

5.1.2 Results for Random Attackers

Random Insiders. We vary parameters in this experiment to select default parameters which are used in the rest of the experiments. We choose the sample length from 500 to 4000 frames, and achieve an EER of 6.79% ($L_s = 500$), 3.12% ($L_s = 1000$), 1.18% ($L_s = 2000$), 0.38% ($L_s = 4000$). Based on the trade-off between the computing overhead and the performance, we select the default sample length to be $2,000$ frames, and create $3,256$ samples for 24 subjects in total. Besides, we randomly select 30 samples per subject for training and use the remaining samples for testing. Note that the training samples are selected from the first few hundred words that were collected in the first batch. This way, we emulate the scenarios where a user enrolls the system and tries to log in the next 7 months. The component length L_c is set to 12 frames, and the number of primitives in the component-feature vocabulary is set to be $K = 200$.

We only utilize the data from the 24 subjects. Each subject is considered as a victim and the remaining 23 subjects are considered as attackers. Results are shown in Figure 10, where each ROC curve represents one subject. We observe that for almost all the subjects, the performance under random attacks are close to ideal, near 100% TPR, 0 FPR. We further calculated an average EER of 1.18% among the 24 subjects.

Random Impostors. In this scenario, the attackers' samples are not part of the negative training samples. Among 24 subjects, we randomly select 12 subjects as impostors and their samples are never part of training sets. We use the remaining 12 subjects as the legitimate users for training. Then we exchange the role of the two subject group. Results shown with ROC curves for all subjects are inFigure 11. The overall EER is 2.26% for all 24 subjects.

5.1.3 Results for Observing Attacks

Observing Attacks by Shoulder Surfing.

We use observing attacks to emulate the extreme case where an attacker happens to observe a user and the system

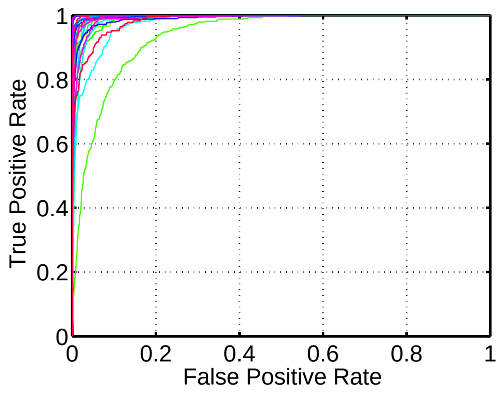

Fig. 11. Performance of all 24 subjects rejecting impostors presents by ROC curves. Each colored curve coresponds to the ROC of one subjct.

asked the attacker to respond to the same challenge. However, it is unlikely for the system to select the same challenge in practice. In this experiment, out of the 24 subjects, we select 4 as victims, and invite 7 attack subjects. 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 in Leap Motion 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.

In the observing attacks, we use the same binary classifiers of the 4 victims trained in the random 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 12 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 average true negative rate of observing attacks is 99.3% and true positive rate is 93.4%. Thus, observing the victim and learning the writing content do not help in impersonating users in MoCRA.

Observing Attacks by a Naive Robot-Arm.

This type of attack is used to emulate a super human and to evaluate his ability to shoulder surfing. We assume the attackers (e.g., a naive robot arm) can precisely record the handwriting of the attacked victims, extract each letter, and synthesize a handwriting by linking the observations of individual letters. Meanwhile, to avoid over inflating the capability of the attacker, we assume that the naive robot arm is not intelligent enough to extract the transition trajectories between letters, nor is it reasonable to record 26×26 transitions between any pairs of 26 letters. To mimic such a naive robot arm attacker in our experiment, we manually slice 26 English letters from subjects' handwriting samples, then we link the letters sequentially to synthesize the handwriting words as attack samples. Note that, this experiment has given favors to the robot arm by assuming it is sufficiently intelligent to slice a handwriting word into a set of individual letters.

In total, we chosen two subjects as the victims, and synthesized 5 paragraphs (238 independent words) of the given document for each victim. Then we apply the binary classifiers of the two victims (trained with subject's genuine handwriting) to these synthesized data for performance evaluation.

The results shows that none of the synthesized sample can pass the SVM-based verification, i.e., the rejection rate is 100%. The results indicate that, perfectly imitating the victim's handwriting of each letter (by recording and replaying) is not enough to fool the MoCRA system. The transition trajectories between letters contain enough handwriting difference between the victim and the attacker, which makes MoCRA resistant to shoulder surfing.

In this experiment, we set the default parameters as below. We select the sample length to be $2,000$ frames, and each sample is formed by randomly selecting as many words as possible, as long as the total sample length L_s is no more than $2,000$ frames. In total, we create $3,256$ samples for 24 subjects and extract sample-level features for re-authentication both on normal scenario and attack scenario. In both scenarios, we randomly select 30 samples per subject for training and use the remaining samples for testing. The component length L_c is set to 12 frames. The number of primitives in the component-feature vocabulary is set to be $K = 200$ (We choose $K = 200$ because it gives a good tradeoff between computation overhead and performance).

5.2 Continous Identification

5.2.1 *Evaluation Metrics*

In this section, we consider the scenario that all users are honest and are part of the known candidate, and we classify a data sample as one of the 24 subjects by using a multi-class SVM algorithm. For this experiment, we only use the normal data collected from the 24 subjects. To quantify the evaluation, we use the 24×24 **confusion matrix**, where the ij-th element A_{ij} is the number of test samples from subject i that were classified as from subject j. The smaller the non-diagonal element in the confusion matrix, the better the identification performance. Based on the confusion matrix, the identification accuracy can be defined as

$$Accuracy(\phi) = \frac{\sum_i A_{ii}}{\sum_i \sum_j A_{ij}}.$$

5.2.2 *Evaluation Results*

Main Results using Default Parameters. The identification results on the 24 subjects are calculated as the confusion matrix. Since not all subjects provide the same number of samples, we normalize the confusion matrix by dividing each row with the total number of testing samples for that subject and display the normalized matrix in Figure 13. The values of the diagonal cells are much larger than the ones of the surrounding cells, which indicates that the samples are correctly identified most of the time.

For instance, subject-1 (the left upper corner block) has 129 samples in total. 124 out of 129 are identified as itself, and only 5 of them are mis-classified: 3 of them are identified as Subject-3, and 2 are identified as subject-21 which results in an accuracy of 96.1%. In total, among all $3,256$ testing samples, $3,172$ samples are correctly identified and the average accuracy is 97.4%.

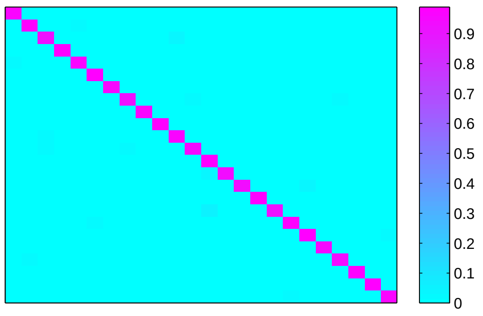

Fig. 13. The confusion matrix for the 24 subjects in the task of identification. The value of each element is encoded into a color as defined by the color bar.

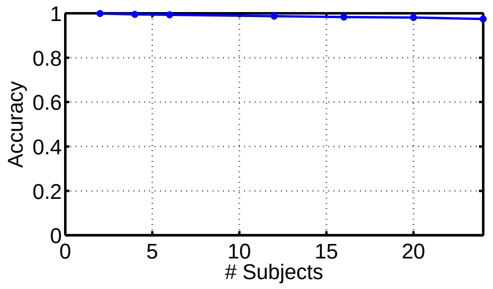

Fig. 14. Identification performance with the increase of subject numbers.

Varying the Number of Candidate Users. In practice, the number of attendees in a remote conference may be small, e.g., 7-10. To evaluate the identification performance as the number of subjects change, we randomly select a subset of the subjects for experiments. The results (the average accuracy) are shown in Figure 14, from which we observe, not surprisingly, that the fewer the subjects, the better the identification accuracy, i.e.,98.3% with 12 users and 99.3% with 6 users. Nevertheless, even with all 24 subjects in the pool, the MoCRA system can correctly identify all subjects with high accuracy.

Varying Parameters. We also conduct experiments to explore the effect of different parameters. In each experiment, we change one parameter, and keep other parameters unchanged, e.g., using the default values mentioned above. Table 2 summarizes the results when varying different parameters.

1) *Length of the data sample* directly affects the usability of the system – the longer the sample, the longer time that takes to input the 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 for a sample length of $L_s = 3,000$ a user has to write 10 more seconds to gain 1% more accuracy compared with the case of $L_s = 2,000$.

2) *The number of training samples.* Our results show that the 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 accuracy of 97.4%. For a system can tolerate an accuracy of 94.6%, 10 samples are enough for training, which requires a training session of 3 minutes. Note that the training session could be reduced further if a lower accuracy is acceptable.

3) *Component length L_c.* We conduct experiments to find the length of components that can represent the coherent features of 3D human handwriting movements. The results in Table 2 show that $L_c = 12$ maps to the best accuracy among the three experiments, and thus we choose each component to be of 12 frames.

4) *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

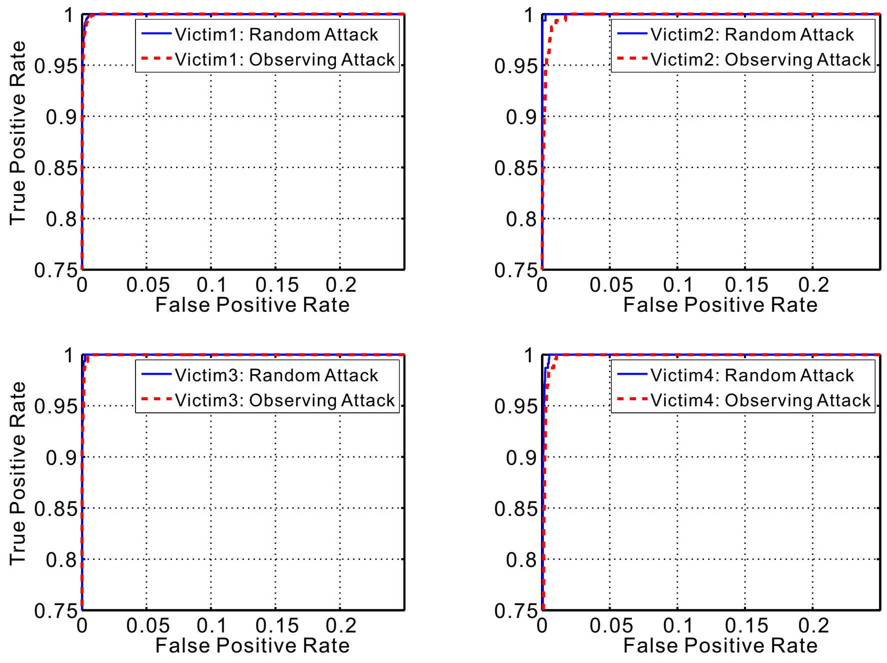

Fig. 12. Authetication performance of the victims under observing attacks presents by ROC curves.

too large, then the size of co-occurrence matrix might 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 calculating features, we choose 200 instead of 400 as the size of the vocabulary.

Histogram vs. Co-occurrence Matrix. We conduct an experiment to justify choosing transition co-occurrence matrix of components as the sample features. In this experiment, we directly use the histogram of the individual (quantized) component features (i.e., the histogram of the component indices), the identification accuracy is only 83.4%, which is much lower than 97.4% that uses the proposed sample features (i.e., co-occurrence matrix). In addition, we combine two features (both histogram and co-occurrence matrix), but the accuracy (i.e., 96.0%) is not as high as the ones using only co-occurrence matrix. Therefore, we conclude that the component-transition statistics can better represent the handwriting style.

6 Related Work

Challenge-response protocols are widely used for user verification over insecure channels. Randomly generated information as the challenge and encrypted or hashed response make the protocols prevent replay attacks [10] and dictionary attacks [5]. O'Gorman *et al.*, without experimental analysis, briefly suggested to create challenge-response protocols with biometrics [37], including speaker verification, keyboard dynamics. Johnson *et al.* [22] proposed to use voice to construct a challenge response method that can verify users without breaching privacy. Their protocol utilizes 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 user. Our work also utilize 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 (i.e., keyboard and mouse) have been proposed for authentication. For instance, keystroke dynamics [33], [34], [40], [43] and patterns of mouse movements or clicks [2], [23] are used for authentication. The gestures that can be measured with new input devices (e.g., touch screens, cameras, or sensors) have attract much attention, too. For instance, wearable accelerometer sensors [13] and smart phones [11] can capture full body gestures (e.g., gaits or strides) for authentication purpose. With the advances in multi-touch screens for smartphones and tablets, gestures of multi-touch (e.g., gestures using multiple fingers at the same time) are studied for authentication. Sae-Bae *et al.* extracted behavior-based biometrics from five-finger gestures and obtained a 90% accuracy [41], and Sherman *et al.* studied the security and memorability on these free form gestures, which are not limited to single or multiple touches [44].

Uellenbeck *et al.* studied the security performance of android system with an unlock pattern called Pass-Go scheme of 3×3 grid size [49]. De LucA *et al.* combined a gesture and how the gesture was entered, and then evaluated the authentication performance [30]. Our work is similar to all the aforementioned work because we all 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 [17], [31] as it is the first low cost depth sensor and provided with several types of data. Leap Motion is another low cost depth sensor but focus on high accuracy and small area depth sensing. Few research has been done on Leap Motion, for instance, the user verification of two gestures that performed under three different device positions [3] and the user identification based on hand shape information [6].

Using motion sensors to capture in-air handwriting was first proposed as a way to enhance text-base passwords [48]. 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 [36]. Their methods 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 a more sophisticated features.

Research on writing style has been used for writer identification that aims at identifying the person who wrote a document or at determining whether multiple documents are

TABLE 2
The identification accuracy (ϕ) on the 24 subjects with various parameters.

Varied Sample Length L_s; Words per Sample wps; Writing Time per Sample $wtps$						
L_s	500	1,000	**2,000**	3,000	4,000	5,000
wps	2.5	5	**10**	15	20	25
$wtps$	4.4s	8.8s	**17.5s**	26.3s	35.1s	43.9s
Accuracy ϕ	83.6%	93.4%	97.4%	98.6%	99.3%	99.3%

Varied # of Training Samples n_T					
n_T	10	20	**30**	40	50
Accuracy ϕ	94.6%	96.8%	97.4%	98.2%	98.4%

Varied Component Length L_c			
L_c	8	**12**	20
Accuracy ϕ	96.7%	97.4%	96.7%

Varied Vocabulary Size K				
K	50	100	**200**	400
Accuracy ϕ	95.0%	97.1%	97.4%	97.9%

written by the same person. Prior work can be divided into text-dependent and text-independent. The text-dependent identification mostly deals with signature verification. Signatures could be obtained online by a digitizing tablet [20] or offline, i.e., scanned images of handwriting [25], [26], [45]. Online-signature in the 3D space [48] was also studied. Our work focuses on text-independent identification. Instead of using the histogram [15], [51] or probability distribution function [7], [8], [42] , we utilize co-occurrence matrix that quantifies the transition information between connecting components, and our results show that the co-occurrence matrix can achieve a better performance than histogram-based methods in our systems.

7 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 the latest motion sensor — Leap Motion controllers — to capture finger movements as a user writes in the air. Our scheme authenticates users based on 'how they write' instead of 'what they write'. We show that the co-occurrence matrix that built on sets of writing components (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 2.26%. In addition, MoCRA can effectively reject skilled attackers that observe victims multiple times, and reject all samples from an emulated motion tracking robot arm.The identification results show that MoCRA can identify a user (e.g., an accuracy of 97.4% for 24 subjects).

Acknowledgment

The authors would like to thank all volunteers for their help collecting data and Carter Bays for improving the paper.

References

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

[2] Ahmed, A. A. E., and Traore, I. A new biometric technology based on mouse dynamics. *IEEE Transaction on Dependable and Security Computing 4*, 3 (2007), 165–179.

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

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

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

[6] Bernardos, A. M., Snchez, J. M., Portillo, J. I., Besada, J. A., and Casar, J. R. A contactless identification system based on hand shape features. *Procedia Computer Science 52* (2015), 161 – 168. The 6th International Conference on Ambient Systems, Networks and Technologies (ANT-2015), the 5th International Conference on Sustainable Energy Information Technology (SEIT-2015).

[7] Biswas, S., and Das, A. K. Content independent writer identification using occurrences of writing styles for bangla handwritings. In *Computer Vision, Pattern Recognition, Image Processing and Graphics, National Conference on* (Los Alamitos, CA, USA, 2011), vol. 0, IEEE Computer Society, pp. 154–157.

[8] Bulacu, M., and Schomaker, L. Text-independent writer identification and verification using textural and allographic features. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 29*, 4 (2007), 701–717.

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

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

[11] Derawi, M., and Bours, P. Gait and activity recognition using commercial phones. *Computers and Security 39* (Nov. 2013), 137–144.

[12] Dimauro, G., Impedovo, S., and Pirlo, G. Component-oriented algorithms for signature verification. *International Journal of Pattern Recognition and Artificial Intelligence 8*, 3, 771–794.

[13] Gafurov, D., Snekkenes, E., and Bours, P. Gait authentication and identification using wearable accelerometer sensor. In *Automatic Identification Advanced Technologies, 2007 IEEE Workshop on* (2007), pp. 220–225.

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

[15] Gordo, A., Fornes, A., Valveny, E., and Llados, J. A bag of notes approach to writer identification in old handwritten musical scores. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems* (2010), pp. 247–254.

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

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

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

[19] Ionu-Alexandru Zaii, tefan-Gheorghe Pentiuc, R.-D. V. On free-hand tv control: experimental results on user-elicited gestures with leap motion.

[20] Jain, A., Griess, F. D., and Connell, S. D. On-line signature verification. *Pattern Recognition 35*, 12 (Dec. 2002), 2963–2972.

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

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

[23] Jorgensen, Z., and Yu, T. On mouse dynamics as a behavioral biometric for authentication. In *Proceedings of the ASIACCS2011* (2011).

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

[25] Kalera, M. K., Srihari, S., and Xu, A. Offline signature verification and identification using distance statistics. *International Journal of Pattern Recognition and Artificial Intelligence 18*, 07 (2004), 1339–1360.

[26] Kamihira, Y., Ohyama, W., Wakabayashi, T., and Kimura, F. Improvement of japanese signature verification by segmentation-verification. *2013 12th International Conference on Document Analysis and Recognition 0* (2013), 379–382.

[27] Kubota, A., Hatori, Y., Matsuo, K., Hashimoto, M., and Koike, A. A study on biometric authentication based on arm sweep action with acceleration sensor. In *Proceedings of Interna-*

tional Symposium on Intelligent Signal Processing and Communication (2006), pp. 219–222.

[28] LEE, H., AND VERMA, B. Binary segmentation algorithm for english cursive handwriting recognition. *Pattern Recognition 45*, 4 (2012), 1306–1317.

[29] LIU, J., ZHONG, L., WICKRAMASURIYA, J., AND VASUDEVAN, V. User evaluation of lightweight user authentication with a single tri-axis accelerometer. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services* (2009), MobileHCI '09.

[30] LUCA, A. D., HANG, A., BRUDY, F., LINDNER, C., AND HUSSMANN, H. Touch me once and i know it's you!: Implicit authentication based on touch screen patterns. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2012), CHI '12, pp. 987–996.

[31] M. GABEL, R. GILAD-BACHRACH, E. R., AND SCHUSTER, A. Full body gait analysis with kinect. In *Proceedings of EMBC 2012* (August 2012), Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC).

[32] MADHVANATH, S., AND BHARATH, A. Hmm-based lexicon-driven and lexicon-free word recognition for online handwritten indic scripts. *IEEE Transactions on Pattern Analysis and Machine Intelligence 34*, 4 (2012), 670–682.

[33] MONROSE, F., REITER, M. K., AND WETZEL, S. Password hardening based on keystroke dynamics. In *Proceedings of the 6th ACM conference on Computer and communications ecurity* (1999), CCS '99, pp. 73–82.

[34] MONROSE, F., AND RUBIN, A. Authentication via keystroke dynamics. In *4th ACM Conference on Computer and Communications Security* (1997), pp. 48–56.

[35] MOTION, L. Leap overview. https://developer.leapmotion.com/documentation/Languages/CSharpandUnity/Guides/Leap_Overview.html.

[36] NIGAM, I., VATSA, M., AND SINGH, R. Leap signature recognition using hoof and hot features. In *Image Processing (ICIP), 2014 IEEE International Conference on* (Oct 2014).

[37] OGORMAN, L. Comparing passwords, tokens, and biometrics for user authentication. In *Proceedings of the IEEE* (2003), vol. 91, IEEE, pp. 2019–2040.

[38] PARK, J., GOVINDARAJU, V., AND SRIHARI, S. N. 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.

[39] PLAMONDON, R., AND SRIHARI, S. N. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Trans. Pattern Anal. Mach. Intell. 22*, 1 (Jan. 2000), 63–84.

[40] REVETT, K. A bioinformatics based approach to user authentication via keystroke dynamics. *International Journal of Control, Automation and Systems 7*, 1 (2009), 7–15.

[41] SAE-BAE, N., AHMED, K., ISBISTE, K., AND MEMON, N. Biometric-rich gestures: A novel approach to authentication on multi-touch devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2012), CHI '12, pp. 977–986.

[42] SCHOMAKER, L., AND BULACU, M. Automatic writer identification using connected-component contours and edge-based features of uppercase western script. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 26*, 6 (2004), 787–798.

[43] SHAVLIK, J., SHAVLIK, M., AND FAHLAND, M. Evaluating software sensors for actively profiling windows 2000 users. In *Proceedings of the Fourth International Symposium on Recent Advances in Intrusion Detection* (2001).

[44] SHERMAN, M., CLARK, G., YANG, Y., SUGRIM, S., MODIG, A., LINDQVIST, J., OULASVIRTA, A., AND ROOS, T. User-generated free-form gestures for authentication: Security and memorability. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services* (2014), MobiSys '14, pp. 176–189.

[45] SOMAYA, A. M. Text-dependent writer identification for arabic handwriting. *Journal of Electrical and Computer Engineering* (2012).

[46] TAPPERT, C. C., SUEN, C. Y., AND WAKAHARA, T. The state of the art in online handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell. 12*, 8 (Aug. 1990), 787–808.

[47] TAPPERT, C. C., SUEN, C. Y., AND WAKAHARA, T. The state of the art in online handwriting recognitionr. In *Pattern Analysis and Machine Intelligence, IEEE Transactions on* (August 1990), vol. 12, pp. 787–808.

[48] TIAN, J., QU, C., XU, W., AND WANG, S. Kinwrite: Handwriting-based authentication using kinect. In *NDSS* (2013).

[49] UELLENBECK, S., DURMUTH, M., WOLF, C., AND HOLZ, T. Quantifying the security of graphical passwords: The case of android unlock patterns. In *Proceedings of ACM CCS* (2013), pp. 161–172.

[50] VIKRAM, S., LI, L., AND RUSSELL, S. Handwriting and gestures in the air, recognizing on the fly. In *In Proceedings of the CHI 2013 Extended Abstracts* (Paris, France, 2013).

[51] WEN, J., FANG, B., CHEN, J., TANG, Y. Y., AND CHEN, H. Fragmented edge structure coding for chinese writer identification. *Neurocomputing 86*, 0 (2012), 45 – 51.

[52] ZHENG, N., PALOSKI, A., AND WANG, H. An efficient user verification system via mouse movements. In *Proceedings of the 18th ACM conference on Computer and communications security* (2011), CCS '11, pp. 139–150.

Jing Tian Biography text here.

Yu Cao Biography text here.

Wenyuan Xu Biography text here.

Song Wang Biography text here.