KinWrite: Handwriting-Based Authentication Using Kinect

Jing Tian*[1], Chengzhang Qu *[2], Wenyuan Xu†[1] and Song Wang‡[1]

[1]Dept. of Computer Science and Engineering, University of South Carolina
[2]School of Computer Science, Wuhan University

Abstract

Password-based authentication is easy to use but its security is bounded by how much a user can remember. Biometrics-based authentication requires no memorization but 'resetting' a biometric password may not always be possible. In this paper, we propose a user-friendly authentication system (KinWrite) that allows users to choose arbitrary, short and easy-to-memorize passwords while providing resilience to password cracking and password theft. KinWrite lets users write their passwords in 3D space and captures the handwriting motion using a low cost motion input sensing device—Kinect. The low resolution and noisy data captured by Kinect, combined with low consistency of in-space handwriting, have made it challenging to verify users. To overcome these challenges, we exploit the Dynamic Time Warping (DTW) algorithm to quantify similarities between handwritten passwords. Our experimental results involving 35 signatures from 18 subjects and a brute-force attacker have shown that KinWrite can achieve a 100% *precision and a* 70% *recall (the worst case) for verifying honest users, encouraging us to carry out a much larger scale study towards designing a foolproof system.*

1 Introduction

Authentication plays a key role in securing various resources including corporate facilities or electronic assets. Naturally, numerous authentication mechanisms have been proposed in the past, and in general they can be divided into three categories: (a) knowledge-based, (b) token-based, (c) biometrics-based. Knowledge-based authentication (e.g., text passwords) has been widely utilized because of its ease of use and ease of update. Unfortunately, text-password-based authentication verifies the ownership of a *text password* instead of a *user* and thus can suffer from password theft [1]—anyone with the text password will pass the authentication. It is also restricted by how much a human can remember—what is hard to guess is often hard to remember [2]. Token-based authentication frees humans from tedious memorizing. It authenticates users by examining their pre-assigned tokens, e.g., physical keys, RFID tags, RSA SecureID tokens, smart cards, smartphones [3], etc. However, such mechanisms are also vulnerable to token theft. Lost or stolen tokens can easily allow anyone pass authentication. Finally, biometrics-based mechanisms that utilize physiological biometrics, e.g., fingerprints, voice, facial and iris patterns, are less likely to suffer from identity theft. However, their applications have received resistance from privacy-savvy users, who worry that they will be tracked, based on their unique physiological biometrics [4].

In this paper, we propose a user-friendly authentication system called *KinWrite* that allows users to choose short and easy-to-memorize passwords while providing resilience to password cracking and password theft. The basic idea is to let a user *write* her password in space instead of *typing* it. Writing in space adds behavioral biometrics to a password (e.g., personal handwriting characteristics) and creates a large number of personalized passwords that are difficult to duplicate. As a result, KinWrite inherits the advantages of both password-based and biometrics-based access control: KinWrite authenticates "who you are" instead of "what you own" or " what you know," and allows users to update their passwords on demand. Hence, stolen passwords, shoulder surfing [5], and user tracking become less of a threat.

To capture in-space handwriting (hereafter *3D-signature*), KinWrite utilizes Kinect [6], which is a low-cost motion input sensor device capable of recording 3D depth information of a human body. Using the depth information, we can detect and track fingertip motion to obtain a corresponding 3D-signature. Kinect is well-suited for this task and can be used in various authentication scenarios including door access control, because it can operate under almost any ambient light conditions, including complete darkness [7]. Verifying a user's identity utilizing 3D-signatures captured by Kinect seems simple yet

*Jing and Chengzhang contributed equally to this work.
†Corresponding Author
‡Emails:{jing9, wyxu, songwang}@cec.sc.edu; quchengzhang@whu.edu.cn

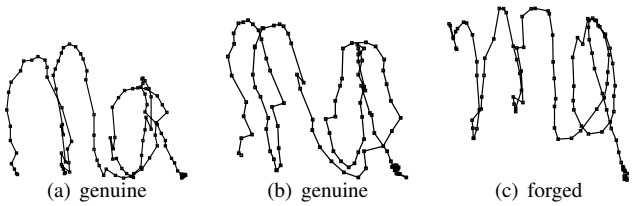
appealing. However, several issues make it a challenging task. First, Kinect is known for its abundant errors and low resolution [8], which may result in distorted 3D-signature samples, as shown in Figure 1. Second, the same users may produce different 3D-signatures over time. Third, the requirement of user-friendly design limits the number of 3D-signatures needed for the initial 'password' enrollment, and thus disqualifies many classification algorithms. Last but not the least, an adversary may try to impersonate a legitimate user. The proposed system has to be able to reject such attempts virtually all the time.

We illustrate the aforementioned challenges in Figure 1: All three signatures are captured using Kinect when a password of 'ma' was written in the space. In particular, Figure 1 (a-b) were written by the same user and Figure 1(c) was generated by an adversary who observed the victim four times and was given the spelling of the password. Although the adversary was able to generate a signature closely imitating the genuine one (shown in Figure 1 (a)) and the two genuine signatures appeared to be different, our KinWrite system is able to correctly identify both genuine signatures and reject the forged one.

KinWrite can verify legitimate users and reject attacks well because it is based on the following intuition. Granted that the shapes of signatures are important, they may change over time and may be learned after being observed visually. In contrast, we believe several spontaneous gestures that are embedded in the movement of in-space handwriting, can characterize each user better and are difficult to imitate, which we will show through our experiments. A user may write letters in different sizes or shapes, but the acceleration at turning points and the transition of consecutive points may not vary much. Thus, to verify a signature, KinWrite examines not only the shape but also several gesture-related features. Lacking a large number of training signatures, KinWrite utilizes Dynamic Time Warping (DTW) to verify signatures, because DTW only requires the storage of one known genuine signature as a template and can accommodate differences in timing between 3D-signatures.

The main contributions of this paper are listed below.

- We propose a behavior-based authentication system (called KinWrite) that combines the benefits of both traditional password-based and biometrics-based schemes. The underlying idea is letting a user write passwords (including short ones) in 3D-space instead of typing them. KinWrite utilizes 3D-signatures as user 'passwords', and it verifies users with the Dynamic Time Warping algorithm.
- We have built a signature capturing system utilizing Microsoft Kinect. With this system, we collected 1180 3D-signatures from 18 users over five months, and with each, we selected up to two different passwords.



(a) genuine (b) genuine (c) forged

Figure 1. The same user may sign the password in 3D space differently while an adversary with knowledge may be able to imitate the shape of the genuine signature closely. Our KinWrite system correctly verified the genuine handwriting (a-b) and rejected the forged one (c).

  In addition, we modelled 5 types of attackers with an increasing amount of information about the passwords and collected 1200 3D-signatures from 18 attackers.
- We evaluated KinWrite using captured 3D-signatures. The results show that KinWrite can effectively verify honest users and is robust to various types of attackers, including the one that observes victims multiple times and is aware of the spelling of the passwords.

Compared with traditional online signatures that uses tablets, KinWrite has the advantage of being contactless, and signing in the 3D-space leaves no traces.

We organize the remainder of the paper as follows. We discuss related work in Section 2. In Section 3, we present the system design requirements and the attack models, introduce Kinect, and overview the KinWrite architecture. Then, we discuss data processing and feature extraction in Section 4, and introduce the Dynamic Time Warping (DTW)-based verification algorithm in Section 5. Finally, we show that KinWrite is effective in verifying users and rejecting various attackers in Section 6 and give our conclusions in Section 7.

2 Related Work

2.1 Authentication

The most widely used, text-based password authentication schemes are known to be susceptible to shoulder surfing, and their security is limited by what people can remember. Graphical passwords are claimed to be a better solution because humans can remember pictures more easily than a string of characters. Recognition-based graphical passwords [2,9] require users to choose their preselected images from several random pictures for authentication, and some schemes [10, 11] have been designed to cope with the problem of shoulder surfing. Another class of graphical passwords asks users to click through several preselected loca-

tions on one image [12]. All those schemes authenticate 'what you know.' In comparison, biometrics-based schemes verify 'who you are.' Traditional biometrics-based schemes utilize physiological biometrics [13], including iris patterns, retina patterns, fingerprints, etc. New approaches utilize behavioral biometrics, such as keystroke dynamics [14, 15] or mouse movements [16, 17], for authentication. KinWrite belongs to the same family of behavioral biometrics. However, most prior behavior-based methods were designed for continuously verifying a user throughout the session as the user operates the keyboard or mouse. KinWrite solves a different problem and targets at authenticating a user once.

Hand-drawn pictures have been proposed as one type of graphical passwords. For instance, Draw-a-Secret (DAS) [18] requires a user to draw a simple picture on a 2D grid, and the user is authenticated if he/she visits the same sequence of grids. KinWrite can also use graphical drawing instead of handwritten signatures as passwords. Nevertheless, in this paper, we focus on studying handwritten signatures. Compared with DAS, whose password space is limited by the number of vertices, KinWrite captures the exact trajectory of a 3D-signaure and thus enables a much larger password space.

2.2 Online Signature Verification

With the development of digital equipment, online signatures have gradually replaced offline signatures (images of signatures) for user identification. For instance, pressure sensitive tablets can record a sequence of 2D signature coordinates as well as pressure. Methods to verify such signatures first extract features from either each sample point or the entire signature [19], and then compare the features against the registered genuine one. The common classification approaches used for comparison include the following: the Bayes classifier [20], Support Vector Machine (SVM) [21, 22], Neural Networks (NN) [23], Hidden Markov Models (HMM) [24, 25], Dynamic Time Warping (DTW) [26, 27]. Several other systems have also been proposed for classification: a pan-trajectory-based verification system [28], verifying based on symbolic features [29], using camera-based signature acquisition [30], or an elastic local-shape-based model [31], etc.

Both KinWrite and online signature utilize behavioral biometrics: handwritten signature. Naturally, the two systems share similarity. However, we believe that 3D-signatures contain richer behavioral information than 2D online signatures captured by tablets. For instance, gesture features are embedded in 3D-signatures, but are difficult to include in 2D online signatures. We envision that 3D-signatures, if done well, can be a good biometric for user authentication.

2.3 Gesture-Based Verification

A few systems have proposed to use hand gestures for user verification. Those systems require users to hold a special device in their hands, such as a phone [32] that captures arm sweep action; a tri-axis accelerometer [33] that captures simple gestures; a biometric smart pen [34] that collects grip strength, the tilt of the pen, the acceleration, etc. KinWrite has the advantage of writing with an empty hand, and such a no-contact method has its advantage to germ conscious users.

2.4 Kinect Application

Kinect, because of its low cost and capability to provide depth and human gesture information, has gained popularity among researchers. It has been used to extract the contour of human body for human identification [35], detect human behavior [36] (e.g, walking, running, etc) utilizing skeleton information, recognize sign language [37], and track a head for augmented reality [38] or fingertips and palms [39]. Kinect is also used in real-time robotics control and building 3D maps of indoor environments [40]. Our system also utilizes the depth information provided by Kinect to track fingertips, but the focus of our work is to verify 3D-signatures.

3 KinWrite Overview

The KinWrite system consists of a Kinect for capturing 3D-signatures, a secure storage for storing abstracts of enrolled 3D-signature templates, and a computing unit for processing data and verifying users. KinWrite, as an authentication system, can be used for various authentication scenarios. Considering that the range of a Kinect sensor is about 0.8m to 4m, KinWrite can work well for office building access control. For instance, a Kinect can be installed at the entrance of a building. To enter the building, a user approaches the Kinect and signs her password towards it. Then, KinWrite will process the captured raw 3D-signature, and authenticate the user by comparing it with the already enrolled genuine 3D-signature.

In this section, we discuss the design requirement of KinWrite, the attack model, the intuition of using a Kinect, and the system architecture.

3.1 System Requirements

Around-the-Clock Use. Similar to most authentication systems, KinWrite is expected to work around the clock, regardless of the weather or lighting conditions.

Rapid Enrollment. Creating new user accounts or updating existing user accounts should be quick, so that users

can set up and reset their 3D-signature passwords easily.

Rapid Verification. The authentication process should require no more than a few seconds.

No Unauthorized Access. One key factor that determines the success of KinWrite is how likely an unauthorized user can pass the authentication. While a bullet-proof system is costly to achieve and may degrade user experiences, KinWrite should ensure that it takes a non-trivial amount of effort for an adversary to impersonate a legitimate user, at least be harder than guessing text-based passwords randomly.

Low False Negative. Users will become frustrated if it takes several attempts to input an acceptable 3D-signature. Thus, KinWrite should have a low false negative, despite several variances that may occur over multiple authentication sessions. For instance, 3D-signatures of the same user may change over time; the distance between a user and a Kinect may vary, affecting the captured 3D-signatures.

3.2 Attack Model

Several mechanisms can be used to protect KinWrite. For instance, opaque panels can be installed at the entrance of a building to block shoulder surfing, and raw 3D-signatures shall never be stored to avoid insider attacks. Nevertheless, we study possible attacks for impersonating legitimate users assuming those protection mechanisms are unavailable.

- **Random Attack**: With no prior knowledge of genuine 3-D signatures, an attacker can randomly sign 3D-signatures and hope to pass the authentication. This is equivalent to a brute force attack against text-based password schemes.
- **Observer Attack**: In an observer attack, an adversary is able to visually observe how a user signs her password once or multiple times and then try to imitate her 3D-signature.
- **Content-Aware Attack**: In a content-aware attack, an adversary knows the corresponding spelling of a legitimate user's 3D-signature, but has not observed how the user signs it in space. The correct spelling can be obtained through social engineering or by an educated guess based on the user's name, hobbies, etc.
- **Educated Attack**: In an educated attack, an attacker is aware of the spelling of a 3D-signature and has observed multiple times how a user signs her password. That is, an educated attack is the combination of an observer attack and a content-aware attack.
- **Insider Attack**: An insider attacker can obtain the spelling of a signature, the corresponding trajectory (i.e., the one shown in Figure 2), and she can observe how a user signs in space. That is, an insider attacker is an educated attacker who knows the signature trajectory. We note signature trajectories are difficult to obtain, since in practice a KinWrite system should never store such information permanently nor display 3D-signatures. Although unlikely to happen, we include this uncommon attack in order to evaluate the performance of KinWrite under extreme attacks.

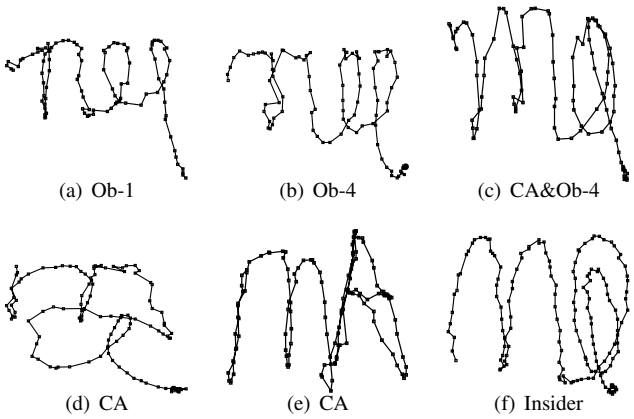
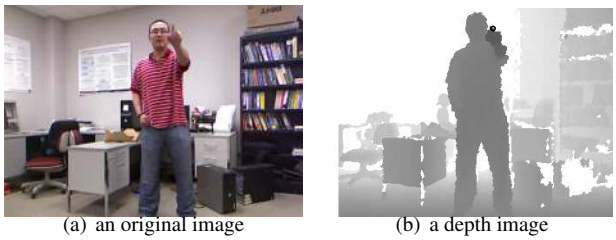


Figure 2. Signatures ('ma') signed by two persons mimicking various attackers. User 1 signed (a)-(c), and user 2 signed (d)-(f). (a) An observer attacker with one observation, (b) an observer attacker with four observations, (c) an educated attacker knowing the spelling and observed four times, (d)-(e) content-aware attackers with known spelling but unaware of the shape of the signature, (f) insider attacker knowing the shape of 3D-signature.

To obtain an intuition on how well the aforementioned attackers can imitate 3D-signatures, we had two users act as attackers and recorded their 3D-signatures when trying to forge the genuine 3D-signature shown in Figure 1 (a). For the first user, we demonstrated the motion of signing 'ma' four times, and then informed him what was written in the space, i.e., we had him act as an observer attacker first then as an educated attacker. For the second user, we asked him to write 'ma' multiple times without demonstrating the motion but gave him the spelling, and then showed the trajectory of the genuine 3D-signature, i.e., we had him act as a content-aware attacker then as an insider attacker. Figure 2 illustrates the signatures signed by the two users, from which we obtain the following intuition: Observing the signing process alone seems to help an attacker to imitate the shape of signatures. However, increasing the number of observations of the signing process does not necessarily improve the forgery in this case. This is encouraging. A larger-scaled experiment that were carried out over five months will be reported in Section 6.



(a) an original image (b) a depth image

Figure 3. The RGB and depth images captured by a Kinect.

3.3 3D-Signature Acquisition Using a Kinect

Basics of a Kinect. A Kinect is a motion input sensing device launched by Microsoft for Xbox 360 and Windows PCs. A Kinect has three sensors: an RGB camera, a depth sensor, and a multi-array microphone. The depth sensor consists of an infrared projector and a monochrome CMOS sensor, which measures the distance between the object and the camera plane at each pixel. With the depth sensor, a Kinect can capture the 3D structure of an object under almost any ambient light conditions [7], including complete darkness. Figure 3 shows example pictures captured by a Kinect: an RGB image of a user who was signing his password and the corresponding depth image. A depth image is shown as a grayscale image, where a darker pixel represents a smaller depth. In this case, the hand of the user is closest to the Kinect.

Why Kinect? We track the hand movement from the captured 3D depth information of the body, with which we can identify the underlying 3D-signatures for verification. This is much more effective than using classical RGB sensors which cannot capture the motion along the depth direction (perpendicular to the image plane). The motion along the depth direction contains important gesture information and can help distinguish 3D-signatures from different subjects. Such information is usually difficult to track from a 2D RGB video, especially when the light is weak or the hand and surrounding background bear a similar color. Before the release of Kinect, other commercialized depth sensors had been used for human posture tracking and gesture recognition [41]. However, these commercialized depth sensors are usually too expensive and only applicable in restricted lab environments [42].

Feasibility of Kinect. Kinect was originally designed for gaming with the goal of capturing the body motion of a player. Will a Kinect suffice for acquiring 3D-signatures? There are two factors determining the applicability of Kinect: the sampling rate and working range. A Kinect can capture 30 frames per second; each frame has a resolution of 240×320 pixels, which is lower than the typical sampling rate (100Hz) in digitizing tablets (used for capturing online signatures). However, the maximum frequencies underlying the human body kinematics are always under 20-30 Hz [43], and the Kinect sampling rate is sufficiently dense for signatures [26]. The working range of the Kinect depth sensor is between 0.8m to 4m (the new version of Kinect can capture the depth from 0.4m to 8m), which works well for the proposed application; For example, at the door of the building, we can allocate an area within the working range of a Kinect, in which a user can move her hand towards the Kinect.

What to Track? One key question is which part of the hand shall be tracked to generate 3D-signatures? For the purpose of modeling, we usually require a signature to be a temporal sequence of points with an infinitely *small* size. Among the options for tracking, e.g., a fingertip, the whole palm or fist, we found the whole palm or fist performs worse than a fingertip because of its relatively large size, with which we cannot find the motion center accurately enough to create a 3D-signature with sufficient spatial resolution. Thus, we track the finger tip, whose corresponding region in the depth map is small, and we can simply take its geometry center as a point on a 3D-signature. As such, we envision that a user will extend his hand in front of his body and use one of his fingers to sign towards the Kinect, as shown in Figure 3 (a). The regions with the smallest value in the Kinect depth map will correspond to the positions of the fingertip most of the time. Note that without a pen, people usually move their fingertips to describe what they want to write. Therefore, the proposed setting of using fingertips for signatures should be natural and spontaneous to most people.

Although Kinect produces depth images that greatly facilitate 3D-signature acquisition, the errors of the depth measurements can be from several millimeters up to about 4cm [40], affecting the accuracy of acquired 3D-signatures. We discuss the mechanisms to address such large measurement errors in Section 4 .

3.4 KinWrite Architecture

Like other authentication systems, authenticating via KinWrite consists of two phases: enrollment and verification. During an enrollment, a user will create an account and enter a few 3D-signatures. Then, KinWrite will first process these genuine 3D-signatures and select one sample as the template for this user. During the authentication phase, a user signs her password towards a Kinect. After preprocessing the newly entered 3D-signature, KinWrite will compare it with the stored template. A match means that the user is genuine, and KinWrite will grant access, otherwise it will deny access.

The computing unit of KinWrite consists of a data preprocessor, a feature extractor, a template selector, and a ver-

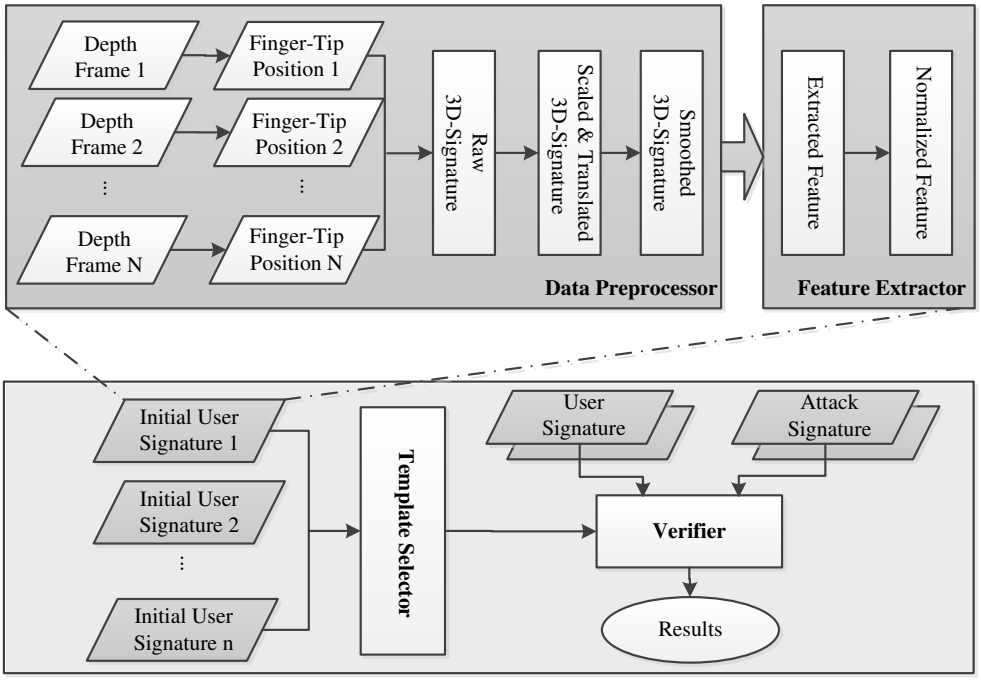


Figure 4. Flow chart of KinWrite. The computing component of KinWrite consists of a data preprocessor, a feature extractor, a template selector, and a verifier.

ifier, as shown in Figure 4. The data preprocessor takes frames captured by a Kinect and outputs a 3D-signature. In particular, the data preprocessor identifies the position of the fingertip that is used for signing a password in the space. By sequentially connecting the fingertips in all frames, KinWrite constructs a raw 3D-signature. Since the size of the raw 3D-signature depends on the distance between a user and the Kinect, we add a data processing step to remove the size difference. Then, a Kalman filter is applied to further reduce the spatial noise in the 3D-signature, and features are extracted for verification.

We discuss the technical details of the data preprocessor and the feature extractor in Section 4, and the template selector and the verifier in Section 5.

4 Data Processing & Feature Extraction

In this section, we describe the techniques to construct a refined 3D-signature from a raw depth image sequence, and discuss feature extraction and its normalization.

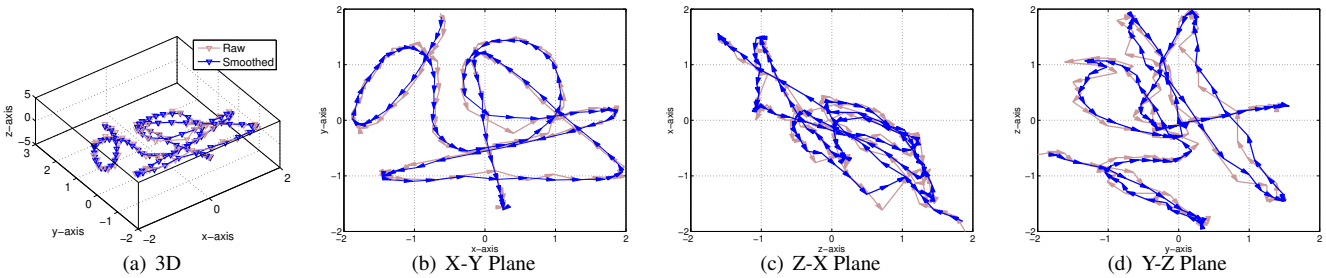
4.1 Data Processing

A data preprocessor performs fingertip localization, signature normalization, and signature smoothing.

4.1.1 Fingertip Localization

Given N frames that capture a 3D-signature, in an ideal case, at each frame t, $t = 1, 2, \cdots, N$, the fingertip (used for the signature) should have the minimum depth. However, in practice, the minimum-depth pixel in a frame may not always correspond to it because of various random noises. To address this issue, we enforce the temporal continuity of the fingertip position in a signature – the fingertip position should only vary in a small range between two consecutive frames. We use the following propagation technique – given the fingertip position $\mathbf{p}^r(t) = (p_x^r(t), p_y^r(t), p_z^r(t))^T$ at the t-th frame, we only search within a small region (40×40 pixels) centered at $\mathbf{p}^r(t)$ in frame $(t+1)$ for the fingertip position. Specifically, in this small region, we choose the pixel with the minimum depth value as $\mathbf{p}^r(t+1)$.

The performance of this frame-by-frame fingertip localization depends highly on a correct fingertip position in the first frame. To ensure the correct initial position, we continue to use the temporal continuity and adopt the following initialization strategy. We choose a small number of the first $K = 3$ frames, and find the pixel with the minimum-depth value in each frame. If they show good temporal continuity (i.e., the identified pixel in a frame is always located in a 40×40 region centered at the pixel identified in the previous frame), we consider them as the fingertip positions in these K frames and process all the other frames



(a) 3D (b) X-Y Plane (c) Z-X Plane (d) Y-Z Plane

Figure 5. A raw 3D-signature (a Chinese character) and the smoothed one using a Kalman filter.

by using the propagation technique described above. Otherwise, we remove the first frame of these K frames and add the next frame to repeat the initialization process until their minimum-depth pixels show the required temporal continuity, which reflects the reliability of the fingertip localization in the initial frames.

4.1.2 Scaling and Translation

By connecting the fingertip points sequentially, we get a raw signature, which is a 3D curve in the $x-y-z$ space. One global feature of a signature is its size, which can be defined by the size of the bounding box around the signature. The size of a signature in the $x-y$ image plane may vary when the distance between the user and the Kinect sensor changes. In addition, users may intentionally sign in a larger or smaller range during different trials, resulting in different sizes of signatures. To achieve a reliable verification, we scale the raw 3D-signatures into a $1 \times 1 \times 1$ bounding box.

To make the different 3D-signatures spatially comparable, we perform a global translation on each signature so that the rear-right corner of its 3D bounding box becomes its origin. Finally, we normalize each position such that it follows a normal Gaussian distribution $\mathcal{N}(0,1)$ over all the frames. We denote the position of the fingertips after the scaling, translation, and normalization to be $\mathbf{p}^s(t) = (p_x^s(t), p_y^s(t), p_z^s(t))^T$.

4.1.3 Signature Smoothing

As shown in Figure 5, the raw 3D-signature obtained by a Kinect is usually highly jagged and noisy. Such jagged signatures are caused by the limited resolution of the Kinect depth sensor. For example, a small area around the fingertip may have similar depths. By selecting the minimum-depth pixel, the above fingertip localization algorithm may not capture the correct fingertip position.

To address this issue, we apply a Kalman filter to smooth the raw 3D-signatures that have been normalized. For simplicity, we smooth the three coordinates of the raw 3D-signature separately. Take the x-coordinate as an example. We denote the prediction of the underlying fingertip position to be $\mathbf{p}(t) = (p_x(t), p_y(t), p_z(t))^T$ at the t-th frame and define the state $\mathbf{x}(t) = (p_x(t), \dot{p}_x(t), \ddot{p}_x(t))^T$ at the t-th frame as a vector of the predicted fingertip position, velocity and acceleration. The state transition of the Kalman filter is then $\mathbf{x}(t) = \mathbf{A}\mathbf{x}(t-1) + \mathbf{w}_x(t)$. Based on the theory of motion under a constant acceleration, we can define

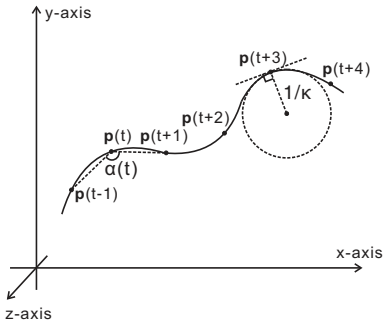
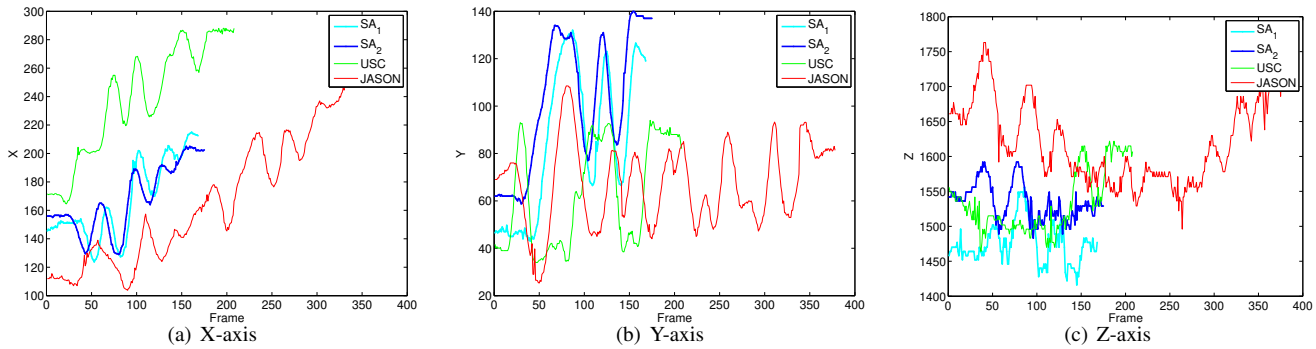


Figure 6. An illustration of path angle and curvature.

$$\mathbf{A} = \begin{bmatrix} 1 & \triangle t & \frac{\triangle t^2}{2} \\ 0 & 1 & \triangle t \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

where $\triangle t$ is the time interval between two consecutive frames. Given the typical rate of 30 frames per second for a Kinect sensor, we have $\triangle t = \frac{1}{30}$ seconds.

For the observation in the x coordinate, we only have the raw fingertip position $p_x^s(t)$ but no velocity or acceleration. Thus, we can write an observation equation for the Kalman filter as $p_x^s(t) = \mathbf{c}\mathbf{x}(t) + v_x(t)$, where $\mathbf{c} = (1\ 0\ 0)$. We model the process noise $\mathbf{w}_x(t)$ and the measurement noise $v_x(t)$ to be zero-mean Gaussian distributions. For the process noise, we choose the same covariance matrix $\mathbf{Q}_x$ for all the frames. More specifically, $\mathbf{Q}_x$ is a 3×3 diagonal matrix with three identical diagonal elements, which equals the variance of acceleration (along x coordinate) estimated from $p_x^s(t)$, $t = 1, 2, \cdots, N$. For the measurement noise,



(a) X-axis (b) Y-axis (c) Z-axis

Figure 7. The position comparison of four 3D-signature samples: two 'SA' 3D-signatures were signed by the same user, 'USC' and 'JASON' were from different users. The two 'SA' 3D-signature samples show a larger degree of similarity than the others.

we choose the time-independent variance v_x as the variance of the fingertip positions (i.e., $p_x^s(t), t = 1, 2, \cdots, N$). Following the same procedure, we set the state-transition and observation equations for y and z coordinates. With the state-transition equation and the observation equation, we use the standard Kalman filter algorithm to calculate a smoothed 3D-signature with refined fingertip positions $\mathbf{p}(t)$, $t = 1, 2, \cdots, N$. Figure 5 shows an example comparing the raw 3D-signature with the smoothed one.

4.2 Feature Extraction

4.2.1 Feature Selection

Based on the refined signature that connects $\mathbf{p}(t)$, $t = 1, 2, \cdots, N$, we extract various features for verification. As discussed earlier, one major advantage of KinWrite is to use simple, easy-to-remember passwords as the basis of 3D-signatures to provide a user friendly authentication method. Given a simple-shape signature, global features, such as the central position and the average velocity, usually do not contain much useful information for distinguishing different signatures. Thus, we extract the following six types of local features at each point and obtain a feature vector of 14 dimensions, as summarized in Table 1.

1. *Position and Position Difference between Frames*. The fingertip position in the t-th frame is denoted as

$$\mathbf{p}(t) = (p_x(t), p_y(t), p_z(t))^T,$$

and the inter-frame position difference is defined as

$$d(t) = \|\mathbf{p}(t+1) - \mathbf{p}(t)\|.$$

2. *Velocity*. The velocity of the position in the t-th frame is defined as

$$\dot{\mathbf{p}}(t) = (\dot{p}_x(t), \dot{p}_y(t), \dot{p}_z(t))^T.$$

3. *Acceleration*. The magnitude of acceleration for the t-th frame is defined as

$$\|\ddot{\mathbf{p}}(t)\|.$$

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

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

5. *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 6.

6. *Curvature*. The last feature extracted for the i-th frame is the log radius of curvature of the signature at $\mathbf{p}(t)$, i.e., $\log \frac{1}{\kappa(t)}$, where $\kappa(t)$ is the curvature in 3D space:

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

where

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

For each frame t, the feature extractor constructs a 14 dimensional feature vector; we denote it as $\mathbf{f}(t)$. Then, for a 3D-signature sample $\mathbf{p}(t)$, $t = 1, 2, \cdots, N$, the feature extractor constructs a sequence of feature vectors $\mathbf{f}(t)$, $t = 1, 2, \cdots, N$. Figure 7 shows some of the features along x, y, and z coordinates for four 3D-signature samples. For ease of reading, we show the feature vectors derived from the raw 3D-signature samples prior to data processing. We observe that the 3D-signature samples from the same user did appear to be similar, which is the basis of verifying users according to their 3D-signatures.

Table 1. The summary of six types ($14-$dimension) of 3D features extracted from smoothed 3D-Signatures.

Type	Features
Positions & Distance	$\mathbf{p}(t)$, $d(t)$
Velocity	$\dot{\mathbf{p}}(t)$
Acceleration	$\|\ddot{\mathbf{p}}(t)\|$
Slope angle	$\theta_{xy}(t)$, $\theta_{zx}(t)$,
Path angle	$\alpha(t)$
Log radius of curvature	$\log \frac{1}{\kappa(t)}$

4.2.2 Feature Processing

In practice, the values of different features may have different ranges, but their relevancy towards the correct verification are not necessarily determined by their ranges. For example, a path angle has a range of $[-\pi, \pi]$ while the position $p_x(t)$ has been scaled to the range of $[0, 1]$. This does not mean that a path angle is 3 times more relevant than a position. Thus, we perform two-step feature processing: normalization and weight selection.

First, we normalize each feature such that it conforms to a normal Gaussian distribution $\mathcal{N}(0, 1)$ over all the frames. Second, we weigh each feature differently to achieve a better performance. To obtain the weight for each feature (dimension), we selected a small set of training samples for each signature (e.g., $n = 4$ samples for each signature), and verified these training samples using the DTW classifier (to be discussed in Section 5) based on one feature (dimension). For each feature (dimension), we obtain a verification rate for each signature, i.e., the percentage of genuine samples in the top $n = 4$ ranked samples, and we simply consider the average verification rate over all signatures as the weight for this feature (dimension). The intuition is that a feature that leads to a higher verification rate should be assigned a larger weight. Our experimental results show that the proposed feature normalization and weighting can substantially improve the verification results.

5 Template Selection and Verification

In this section, we elaborate on algorithms to verify users, based on their 3D-signatures.

5.1 Why Dynamic Time Warping

A good verification algorithm should perform accurately without requiring a large number of training samples, because from the usability perspective, it is unpleasant to collect a large number of training samples when a user enrolls herself.

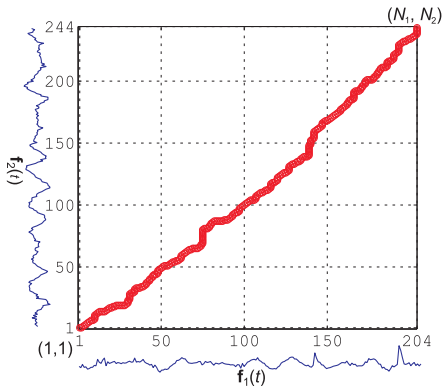


Figure 8. An illustration of DTW.

Hidden Markov Models (HMM) are well-known statistical learning algorithms used in classical signature-based verification systems and have shown good verification accuracy. However, HMM usually requires a large training set (i.e., representative signature samples) to construct an accurate model. With the usability constraints, it is difficult to perform well, as has been validated with our experiments. Thus, we use Dynamic Time Warping (DTW), where one good template is sufficient for verification.

We use DTW to quantify the difference between two 3D-signature samples. Instead of directly calculating the feature difference in the corresponding frames, DTW allows nonrigid warping along the temporal axis. To some degree, time warping can compensate the feature difference caused by the signing speed. For instance, a user may sign her 3D-signature slowly one day and quickly another day. Given two 3D-signature samples, we denote their feature vectors as $\mathbf{f}_1(t)$, $t = 1, 2, \cdots, N_1$ and $\mathbf{f}_2(s)$, $s = 1, 2, \cdots, N_2$, and construct a $N_1 \times N_2$ distance matrix $\mathbf{D}$ with an element $d_{ts} = \|\mathbf{f}_1(t) - \mathbf{f}_2(s)\|$, $t = 1, 2, \cdots, N_1$, $s = 1, 2, \cdots, N_2$. DTW finds a non-decreasing path in $\mathbf{D}$, starting from d_{11} and ending at $d_{N_1 N_2}$, such that the total value of the elements along this path is minimum. This minimum total value is defined as the DTW distance between the two 3D-signature samples; we denote it as $d(\mathbf{f}_1, \mathbf{f}_2)$. Figure 8 illustrates such an example.

5.2 Template Selection

Utilizing DTW as the verification algorithm, during the enrollment phase for a user u, we simply choose *the* most representative 3D-signature sample $\mathbf{f}^u$ from the training set, which we call the template (3D-signature) of the user u. With this template, we can verify a test 3D-signature sample $\mathbf{f}$ of the user u by evaluating their DTW distance $d(\mathbf{f}^u, \mathbf{f})$: If the DTW distance is larger than a threshold d_T, the verification fails. Otherwise, the verification succeeds.

How well KinWrite performs is determined by the

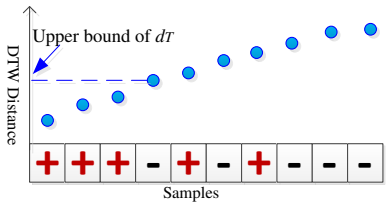


Figure 9. An illustration of threshold selection.

choice of the template. To select a template for each user, we use a distance-based strategy and consider only her own training samples. In this strategy, given n training 3D-signature samples $\mathbf{f}_1, \mathbf{f}_2, \cdots, \mathbf{f}_n$ for a user u, we calculate the pairwise DTW distance $d(\mathbf{f}_i, \mathbf{f}_j)$, $i, j = 1, 2, \cdots, n$, and choose the template that has the minimum total DTW distance to all these n samples, i.e.,

$$\sum_{j=1}^{n} d(\mathbf{f}^u, \mathbf{f}_j) \leq \sum_{j=1}^{n} d(\mathbf{f}_i, \mathbf{f}_j), i = 1, 2, \cdots, n. \quad (2)$$

5.3 Threshold Selection

Another important issue for verifying a 3D-signature sample is threshold selection. The 3D-signatures from different users may have different thresholds, and therefore we select a threshold d_T for each user. Since most verification systems prefer to reduce unauthorized accesses to minimum, we aim to select a threshold that leads to a zero false positive rate for the training samples, i.e., training signature samples that are not from a user u cannot pass the verification. During the enrollment phase, we calculate the DTW distance between the template of a user u and all the M training samples (from all the users), and sort them. We find the first training sample in the sorted list that is not from the user u. Then, the DTW distance between this sample and the template of the user u is the upper bound of d_T, and we select a d_T that is smaller than the upper-bound to achieve a higher level of security. Figure 9 shows an example of $M = 10$ training samples. The x-axis gives the indices of the training samples and the y-axis is their DTW distance to the template of the user u. Along the x-axis, the samples that are labeled '+' are genuine training samples from the user u while samples labeled '-' are training samples from other users. In this case, the upper-bound of d_T is the distance between the template and the first '-' sample along the x-axis. In the experiment, we tried various threshold values to construct the precision-recall curve and the ROC curve, and hence to evaluate the system performance comprehensively.

6 Experiment and Evaluation

In this section, we present experiment results to justify the proposed verification method.

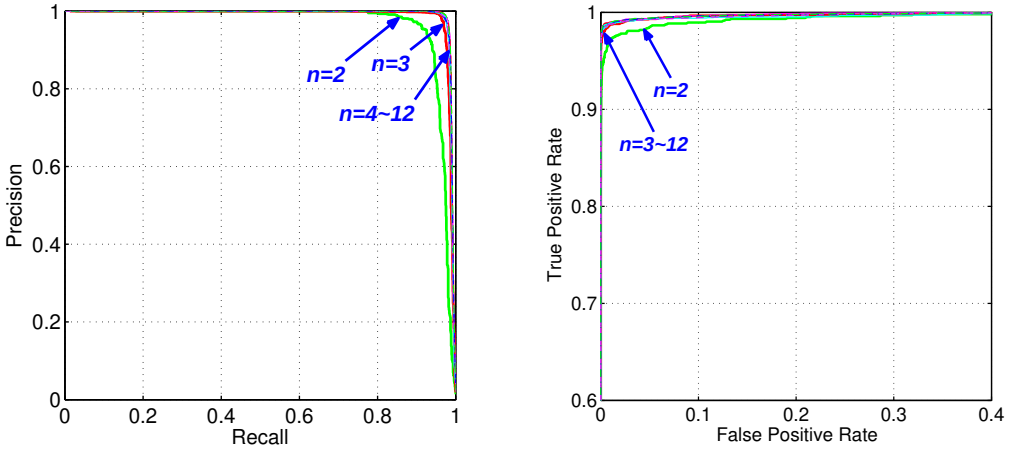
6.1 Data Acquisition

We use the Microsoft Kinect for data collection. In our collected data, each sample is a short video clip that captures the motion of signing one 3D-signature sample. The length of the video clip may vary for each sample, but typically is in the range of $[2, 12]$ seconds. We set the frame rate to the maximum allowed value (i.e., 30 frames per second), and set the resolution of the depth image to 240×320 pixels. The distance between the user and the Kinect was not fixed, but was in the range of $[1.5, 2.5]$ meters. We alternated three Kinect sensors for data collection and did not differentiate samples collected by different Kinect sensors to validate that our algorithm is insensitive to individual Kinect sensors.

In total, we studied 18 users, allowing each user to enroll up to two different 3D-signatures (e.g., 'ma' and 'Bry' are from the same user). In total, these users provided 35 different 3D-signatures, which we call signatures hereafter. For each signature, we collected 18 to 47 3D-signature *samples* over a period of five months so that we could capture the possible 3D-signature variation over time. We collected fewer samples for some signatures because the users were not always available over the entire five months of data collection. In total, we collected 1180 genuine 3D-signature samples for 35 signatures, and hereafter we call these samples *the genuine samples*.

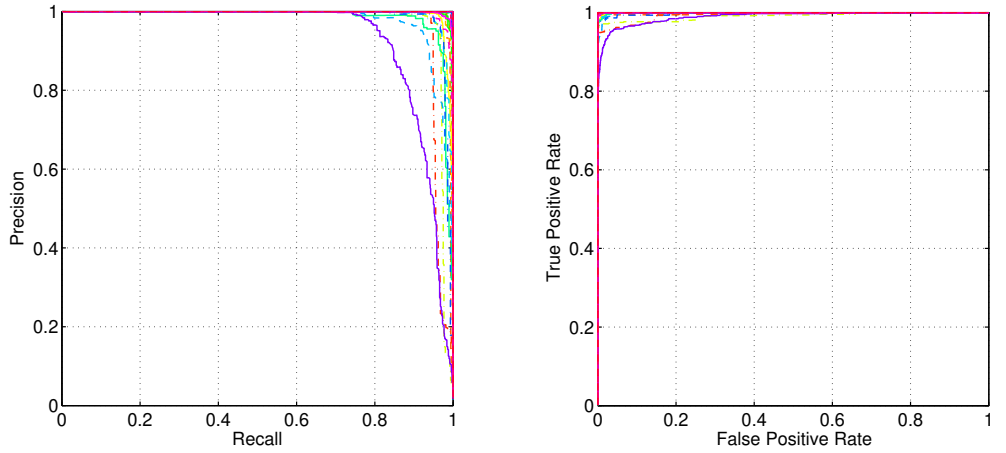
We further collected attack data to evaluate the security performance of KinWrite against impersonation attempts. In particular, we chose four signatures as the 'victims', and for each victim, we collected five types of attack samples that simulate five different attack models.

- **CA**. We chose six attackers to launch content-aware attacks. We gave them the spelling of the victims' passwords, without any hint of the passwords' geometry or shape. Then, each of these six attackers produced 10 forged 3D-signature samples for each victim. In total we collected $6 \times 10 \times 4 = 240$ CA attack 3D-signature samples.
- **Ob-1**. We selected a different group of 12 attackers to perform observer attacks. Each of them watched the signing process of each victim *once* and then produced five forged 3D-signature samples. Given the four victims, we collected $12 \times 5 \times 4 = 240$ Ob-1 attack samples.
- **Ob-4**. The same 12 attackers continued to observe the signing process of each victim three more times



(a) Precision-Recall curves (b) ROC Curves

Figure 10. Training performance of KinWrite with different n, the number of training samples for each signature. For ROC curves, the range of x-axis is $[0, 0.4]$ and the range of y-axis is $[0.6, 1]$.



(a) Precision-Recall Curves (b) ROC Curves

Figure 11. The performance of KinWrite (by signatures) in normal cases. Each colored curve indicates the performance of verifying one signature.

(in total *four times*) and then produced five forged 3D-signature samples. In total, we collected $12 \times 5 \times 4 = 240$ Ob-4 attack 3D-signature samples.

- **CA-Ob4**. After collecting the Ob-4 samples, we gave the same 12 attackers the spelling of the passwords. Then, each of these 12 attackers produced five new forged 3D-signatures for each victim. In total we collected $12 \times 5 \times 4 = 240$ CA-Ob4 attack 3D-signature samples.
- **Insider**. We told six attackers the spelling, showed them three representative 3D-signature samples of each victim (printout on papers), and let them watch the signing process of each victim *once*. Each of these six attackers then produced 10 forged 3D-signature samples for each victim. This way, we collected $6 \times 10 \times 4 = 240$ Insider attack 3D-signature samples in total.

Combining all five types of samples, we collected $240 \times 5 = 1,200$ attack 3D-signature samples. From CA samples to Insider samples, the attackers gained an increasing amount of prior knowledge about the victims, representing a broad range of security threats to KinWrite.

6.2 Evaluation Metrics

We adopted standard ROC curves and precision-recall curves to evaluate the performance of KinWrite. For each threshold d_T, we tried m rounds. For round i, the classification results can be divided into the following four cate-

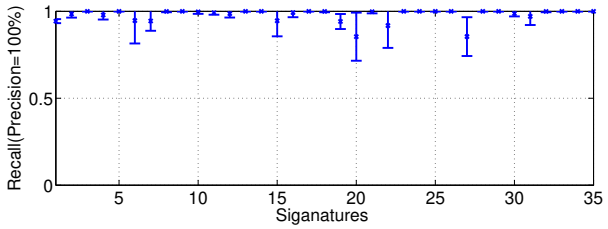


Figure 12. The performance of KinWrite in normal cases: the averages and standard deviations of the achievable recall at a 100% precision.

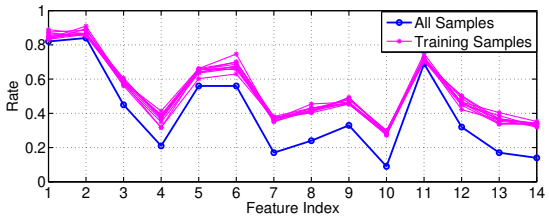


Figure 13. The impact of the sample size on the feature weight selection: The weights obtained over a randomly selected training set with 4 samples are similar to the one obtained over all samples.

gories: tp_i, the number of true positives; tn_i, the number of true negatives; fp_i, the number of false positives, and fn_i, the number of false negatives.

Precision is the percentage of honest users out of all the users that have passed verification, and it reflects how cautious the system is to accept a user. A secure system should have a precision of 100% and will only let honest users pass the verification. Formally,

$$Precision = \frac{\sum_{i=1}^{m} tp_i}{\sum_{i=1}^{m} tp_i + \sum_{i=1}^{m} fp_i}.$$

Recall is the number of true positives over the sum of true positives and false negatives. It quantifies the fraction of honest users that have been granted access out of all honest users, and it affects the user experience. Formally,

$$Recall = \frac{\sum_{i=1}^{m} tp_i}{\sum_{i=1}^{m} tp_i + \sum_{i=1}^{m} fn_i}.$$

A recall of 100% indicates that an honest user can always pass the verification at her first trial. A recall of 50% indicates that an honest user has a 50% probability of gaining access. On average it takes 2 trials to pass the verification.

ROC curve stands for receiver operating characteristic curve and is a plot of true positive rate (TPR) over false positive rate (FPR). An ideal system has 100% TPR and 0% FPR, i.e., all honest users can pass the verification while none of the attackers can fool the system.

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

By varying the threshold d_T, we can achieve varied precision, recall, TPR and FPR values with which we can draw precision-recall curves and ROC curves.

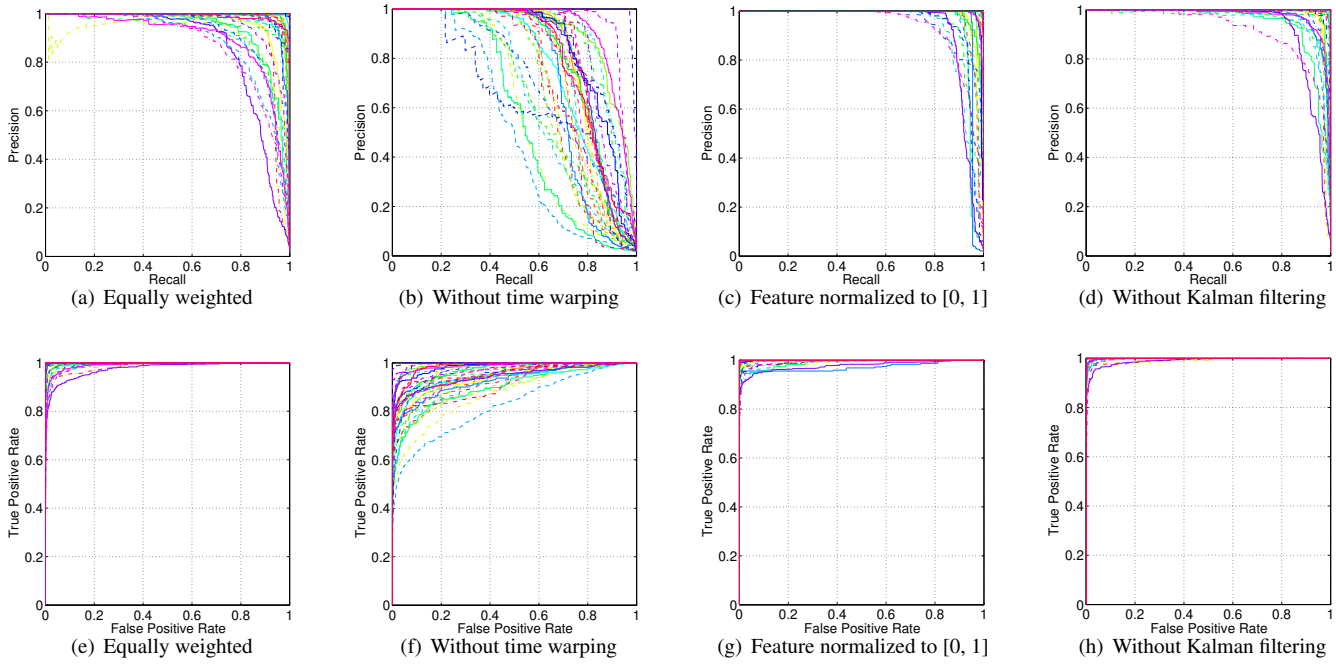
6.3 Evaluation Results

We performed two sets of experiments utilizing the 3D-signature samples collected over five months. The first set of experiments studied the performance of KinWrite in a normal scenario, where honest users want to authenticate themselves. The second set of experiments studied the performance of KinWrite under various attacks.

6.3.1 Normal Case Performance

In our first set of experiments, we divided the genuine samples into two sets: a training set and a test set. We randomly selected a subset of n genuine samples for each of the 35 signatures as their training samples and let the remaining samples be the test set. KinWrite selected a template for each signature from the training samples, and then used the test samples to evaluate the verification performance. To study the statistical performance of KinWrite, we conducted 30 rounds of random folding, where for each round, a different set of n samples were selected as training samples. We reported the performance over the 30 rounds of experiments and for all 35 signatures.

Training Size. We first conducted experiments to evaluate the impact of training size n on the verification performance. In each round, we randomly selected n samples as the training samples. In total, $M = 35 \cdot n$ training samples were selected for all signatures. For each signature, our template selector chose one template and sorted all M training samples according to the DTW distances, as shown in Figure 9. By varying the threshold d_T, we obtained a ROC curve and a precision-recall curve. As we tried n in the range of $[2, 12]$, we obtained a set of ROC curves and a set of precision-recall curves, as shown in Figure 10, where performance for each value of n is over 30 rounds and 35 signatures. We observe that the performance is not too sensitive to the selection of n as long as $n > 2$, and when $n > 2$, KinWrite can almost achieve a precision of 100%



(a) Equally weighted (b) Without time warping (c) Feature normalized to [0, 1] (d) Without Kalman filtering

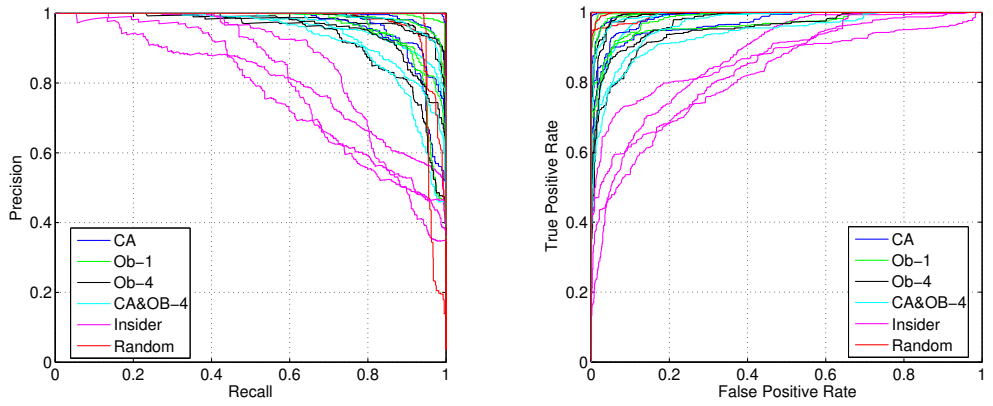
(e) Equally weighted (f) Without time warping (g) Feature normalized to [0, 1] (h) Without Kalman filtering

Figure 14. The performance comparison of various methods (by signatures) in normal cases. Each colored curve indicates the performance of verifying one signature. The top row shows the precision-recall curves, and the bottom one shows the ROC curves.

and a recall of 90%. Thus in the remainder of our experiments, we chose $n = 4$.

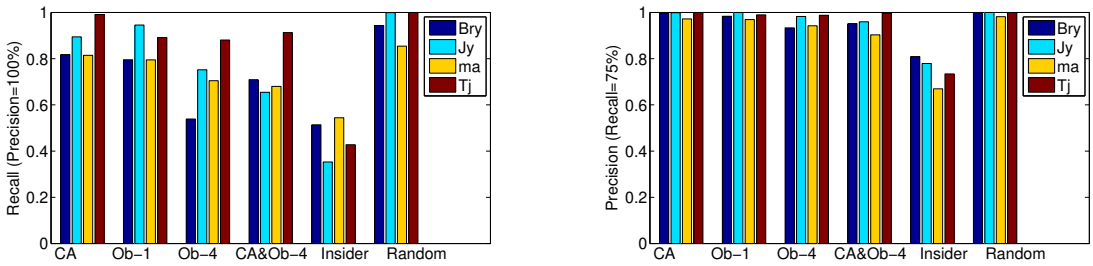
KinWrite Performance. Figure 11 shows the test performance (ROC and precision-recall curves) of the 35 signatures when the training sample size is 4. As before, we tried 30 rounds of random folding for each signature, and each curve represents the performance result averaged over all 30 rounds for a signature. Our experimental results show that given a requirement of a 100% precision, we can achieve at least a 70% recall or a 99% recall on average. Assuming that 3D-signature samples are independent, the probability that an honest user passes verification is about 70%. Since the number of successes of n trials can be considered as a Binomial distribution, the average number of trials for a user to pass the verification is $\frac{1}{70\%}$. In Figure 12, we show the averages of maximum achievable recall for each signature when the precision was 100%, from which we observed the following: 17 out of 35 signatures can achieve a 100% recall; 13 signatures achieved a recall higher than 95%, and the rest achieved a recall higher than 85%. The results suggest that as with text passwords, some 3D-signatures are better then others. Nevertheless in our experiments, KinWrite can verify an honest user by 1.4 trials on average without false positives.

Feature Weight Selection and Its Impact. Since the relevancy level of each feature (dimension) varies for verifying a 3D-signature, we weigh each feature differently in order to achieve a high verification performance. Weights are selected based on the verification rate obtained purely on a small training set. To understand how sensitive weight selection is to training samples, we calculated weights when different sets of the samples were used. In the first set of experiments, we randomly selected 4 samples from each signature as the training samples. In total, $M = 140$ training samples were selected for all signatures. For each signature, we calculated the DTW distance between training samples based on only a single feature. We chose the weight of that feature as the average verification rate of all 35 signatures (i.e., the percentage of true samples out of the top-ranked 4 samples, when verifiying each signature using all M samples). We repeated the process 10 rounds by selecting 10 different training sets for each signature, and depicted the derived weights in Figure 13. We observed that the weights obtained over training sample sets are similar to each other. We also calculated the weights by considering all the available samples (shown in Figure 13). The resulting weights are similar to the ones derived based on training sets, suggesting that weight selection over a small training set suffices.



(a) Precision-Recall curves (b) ROC Curves

Figure 15. The average performance (by signatures) in various attack scenarios.



(a) Achievable recall at a 100% precision (b) Achievable precision at a 75% recall

Figure 16. The performance (by signature) in various attack scenarios.

To evaluate the impact of weighted features on verification performance, we modified KinWrite so that all 14 dimensions of the features were equally weighted. Figure 14 (a, e) show the verification performance on all 35 signatures of this modified KinWrite. The results demonstrate that the proposed weighting features can improve the verification performance.

The Role of Dynamic Time Warping. The proposed DTW allows nonrigid warping along the temporal axis when measuring the difference between two signatures. To understand the impact of nonrigid warping on the verification performance, we defined the difference between two signatures (in the form of features) $\mathbf{f}_1(t)$, $t = 1, 2, \cdots, N_1$ and $\mathbf{f}_2(t)$, $t = 1, 2, \cdots, N_2$ as follows. We re-sampled the signature features so that they had the same length, e.g., $N = 50$ points, and then calculated the Euclidean distance between the two signature feature vectors. Figure 14 (b, f) shows the verification performance (on all 35 signatures) when using this difference metric without warping along the temporal axis. The results show that the use of nonrigid warping in DTW can substantially improve the verification performance.

Impact of Kalman Filter and Feature Normalization. We conducted experiments to justify the choice of Kalman filter and feature normalization. First, we modified our KinWrite so that the Kalman filter was not included, or a different feature normalization method was used by the data preprocessor, and then we conducted the experiment as before. Figure 14 (c, g) show the verification performance on all 35 signatures when features were normalized linearly to the range of [0, 1]. The results show that the proposed feature normalization method based on $\mathcal{N}(0, 1)$ distribution leads to a better performance. Figure 14 (d, h) show the verification performance on all 35 signatures when the signatures were not smoothed by the proposed Kalman filter. From the results, we can conclude that the use of a Kalman filter can improve the verification performance.

6.3.2 Attack Performance

In the attack experiments, we evaluated how robust KinWrite is against various types of attackers. We selected four signatures as the victims with the spelling being "Bry", "Jy", "ma", and "Tj", respectively. We considered the other 31 signatures acquired for the first set of experiments as ran-

dom attackers and collected forged data for all types of attackers described in Section 6.1. Similar to the first set of experiments, we divided samples into two sets: a training set and a test set. For each type of attack, the training set of a victim signature consists of 4 randomly chosen samples from each victim signature and this type of attacker samples. The test set contains the rest of the samples from all victims and this type of attacker.

For each type of attacker, we performed 30 rounds of random folding. We averaged precision-recall curves and ROC curves over 30 rounds for each victim and showed performance results in Figure 15, where each type of attacker has four identical-colored curves with each corresponding to one of the four victims. The results show that KinWrite can with a high probability reject random attackers. 'Random' indicates a brute force attack– an attacker who has no clue about the 3D-signatures and signs random texts hoping to pass the verification. The results suggest that KinWrite is robust against brute force attacks. For other types of attacks, Kinwrite did not perform as well as for the random attacks, which is not surprising since these types of attackers had partial information about the signatures.

In Figure 16 (a), we summarized the maximum achievable recall for each victim under all attack models, when the precision is required to be 100%. This figure provides insight on the trade-off between security and usability. For instance, operator #1 may aim to tune KinWrite so that it can prevent random attackers from passing verification with a high confidence, while operator #2 may tune KinWrite to block insider attackers. As a result, on average fewer trials are required for an honest user to pass verification in the first system than in the second system. Figure 16 (b) shows the precision when the recall was 75%. This figure illustrates how easily an attacker can pass verification, when an operator decides to tune KinWrite so that users can pass verification by $\frac{1}{75\%}$ trials on average. In our experiments, we observed that CA attackers, Ob attackers, and CA&Ob-4 attackers had a slightly higher chance to pass verification than random attackers, but KinWrite would reject all of them (5 types) with a probability of 97% on average and reject insider attackers with a 75% probability on average.

In addition, the results suggest that the choice of signature affects the performance of KinWrite, since some signatures are more robust to shoulder surfing than others. For instance, the signature `'Tj'` is the hardest to imitate among all four signatures, and watching the signing motion multiple times did not improve the imitation. In comparison, the signature `'Bry'` was the easiest to mimic, and observing multiple times helped. The feedback from 'attackers' reveals the reason: `'Bry'` was signed much more slowly than `'Tj'`. Hence, the slow motion of `'Bry'` made imitation easier while the fast motion and the ambiguous shape of the signature `'Tj'` made the task difficult. Interestingly, after we gave the spelling of the signatures `'Bry'` and `'Tj'` to the Ob-4 attackers, they could no longer mimic as well as they used to, because they started to write the text in their own style instead of purely emulating the signature motion.

In summary, our experiments show that KinWrite can reject most attackers with a high probability. Even with a strong attacker (i.e., an insider attacker), KinWrite perform gracefully. In real application scenarios, many of these attacks, especially Insider attacks, can be prevented by physical protection or by a good system design. For instance, knowing the exact shape of a 3D-signature will increase the chances of a successful attack, and thus KinWrite does not display the signed 3D-signature in real time and only stores the normalized feature vectors of templates.

7 Conclusion

We have designed a behavior-based authentication system called KinWrite that can be used for building access control. By letting users sign their passwords in 3D space, we turned short and easy-to-crack passwords into behavioral biometrics, i.e, 3D-signatures. KinWrite utilizes Kinect, a low-cost motion input sensor, to capture fingertip movement when a user signs her password in space, and constructs a 3D-signature. To verify a user, based on her 3D-signatures that may change over time, we extracted features that are likely to contain personal gesture information, and we used Dynamic Time Warping to calculate the similarity between samples. One advantage of using DTW is that KinWrite only needs to store one template for each user.

To evaluate the performance of KinWrite, we collected 1180 samples for 35 different signatures over five months. In addition, we modelled 5 types of attackers who tried to impersonate a legitimate user, and collected 1200 3D-signature samples from 18 'attackers'. The evaluation results obtained using these samples show a 100% precision, and a 99% recall on average in the presence of random attackers, e.g., an attacker trying to impersonate a legitimate user in a brute force manner; a 100% precision and a 77% recall on average for all attackers. These results suggest that KinWrite can deny the access requests from all unauthorized users with a high probability, and honest users can acquire access with 1.3 trials on average.

Acknowledgement

The authors would like to thank all volunteers for their help collecting data and Carter Bays for improving the paper. This work has been funded in part by NSF CNS-0845671, NSF GEO-1124657, AFOSR FA9550-11-1-0327, NSF-1017199, and ARL W911NF-10-2-0060.

References

[1] Y. Zhang, F. Monrose, and M. K. Reiter, "The security of modern password expiration: an algorithmic framework and empirical analysis," in *Proceedings of the 17th ACM conference on Computer and communications security*, 2010, CCS '10, pp. 176–186.

[2] X. Suo, Y. Zhu, and G. S. Owen, "Graphical passwords: A survey," in *Proceedings of the 21st Annual Computer Security Applications Conference*, 2005, ACSAC '05, pp. 463–472.

[3] J. Cornwell, I. Fette, G. Hsieh, M. Prabaker, J. Rao, K. Tang, K. Vaniea, L. Bauer, L. Cranor, J. Hong, B. McLaren, M. Reiter, and N. Sadeh, "User-controllable security and privacy for pervasive computing," in *IEEE Workshop on Mobile Computing Systems and Applications (HotMobile)*, Feb 2007, pp. 14–19.

[4] NSTC Subcommittee on Biometrics and Identity Management, "Privacy & biometrics: Building a conceptual foundation," pp. 1–57, 2006.

[5] F. Tari, A. A. Ozok, and S. H. Holden, "A comparison of perceived and real shoulder-surfing risks between alphanumeric and graphical passwords," in *Proceedings of the second symposium on Usable privacy and security*, 2006, SOUPS '06, pp. 56–66.

[6] "Kinect," http://www.xbox.com/en-US/KINECT.

[7] Z. Zhang, Chen Z, J. Shi, F. Jia, and M. Dai, "Surface roughness vision measurement in different ambient light conditions," *Int. J. Comput. Appl. Technol.*, vol. 39, no. 1/2/3, pp. 53–57, Aug. 2010.

[8] K. Khoshelham, "Accuracy analysis of kinect depth data," *GeoInformation Science*, vol. 38, no. 5/W12, pp. 1, 2010.

[9] R. Dhamija and A. Perrig, "Deja vu: a user study using images for authentication," in *Proceedings of the 9th conference on USENIX Security Symposium*, Aug. 2000, SSYM'00.

[10] S. Wiedenbeck, J. Waters, L. Sobrado, and J-C. Birget, "Design and evaluation of a shoulder-surfing resistant graphical password scheme," in *Proceedings of the working conference on Advanced visual interfaces*, 2006, AVI '06, pp. 177–184.

[11] A. Forget, S. Chiasson, and R. Biddle, "Shoulder-surfing resistance with eye-gaze entry in cued-recall graphical passwords," in *Proceedings of the 28th international conference on Human factors in computing systems*, 2010, CHI '10, pp. 1107–1110.

[12] L. D. Paulson, "Taking a graphical approach to the password," *Computer*, vol. 35, 2002.

[13] N. K. Ratha, J. H. Connell, and R. M. Bolle, "Enhancing security and privacy in biometrics-based authentication systems," *IBM Syst. J.*, vol. 40, no. 3, pp. 614–634, 2001.

[14] 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.

[15] 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 security*, 1999, CCS '99, pp. 73–82.

[16] 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*, 2011, CCS '11, pp. 139–150.

[17] 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.

[18] I. Jermyn, A. Mayer, F. Monrose, M. K. Reiter, and A. D. Rubin, "The design and analysis of graphical passwords," in *Proceedings of the 8th conference on USENIX Security Symposium*, Aug. 1999, vol. 8 of *SSYM'99*, pp. 1–14.

[19] J. Richiardi, H. Ketabdar, and A. Drygajlo, "Local and global feature selection for on-line signature verification," in *Proceedings of the 8th International Conference on Document Analysis and Recognition*, 2005, ICDAR '05, pp. 625–629.

[20] J. Fierrez-Aguilar, L. Nanni, J. Lopez-Pe nalba, J. Ortega-Garcia, and D. Maltoni, "An on-line signature verification system based on fusion of local and global information," in *Proceedings of the 5th international conference on Audio- and Video-Based Biometric Person Authentication*, 2005, AVBPA'05, pp. 523–532.

[21] H. Byun and S-W. Lee, "Applications of support vector machines for pattern recognition: A survey," in *Proceedings of the First International Workshop on Pattern Recognition with Support Vector Machines*, London, UK, 2002, SVM '02, pp. 213–236, Springer-Verlag.

[22] N. Cristianini and J. Shawe-Taylor, *An introduction to support Vector Machines: and other kernel-based learning methods*, Cambridge University Press, New York, NY, USA, 2000.

[23] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 1998.

[24] J. Fierrez, J. Ortega-Garcia, D. Ramos, and J. Gonzalez-Rodriguez, "Hmm-based on-line signature verification: Feature extraction and signature modeling," *Pattern Recognition Letters*, vol. 28, pp. 2325–2334, 2007.

[25] D. Muramatsu and T. Matsumoto, "An hmm on-line signature verifier incorporating signature trajectories," in *Proceedings of the 7th International Conference on Document Analysis and Recognition*, 2003, vol. 1 of *ICDAR '03*, pp. 438–442.

[26] A. Jain, "On-line signature verification," *Pattern Recognition*, vol. 35, no. 12, pp. 2963–2972, Dec. 2002.

[27] A. Kholmatov and B. Yanikoglu, "Identity authentication using improved online signature verification method," *Pattern Recognition Letters*, vol. 26, no. 15, pp. 2400–2408, Nov. 2005.

[28] T. Ohishi, Y. Komiya, and T. Matsumoto, "On-line signature verification using pen-position, pen-pressure and pen-inclination trajectories," in *Proceedings of the International Conference on Pattern Recognition*, 2000, vol. 4, pp. 547–550.

[29] 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.

[30] D. Muramatsu, K.K. Yasuda, and T. Matsumoto, "Biometric person authentication method using camera-based online signature acquisition," in *Proceedings of the 2009 10th International Conference on Document Analysis and Recognition*, 2009, ICDAR '09, pp. 46–50.

[31] V. S. Nalwa, "Automatic on-line signature verification," in *Proceedings of the IEEE third Asian Conference Computer Vision*, 1997, pp. 215–239.

[32] A. Kubota., Y. Hatori., K. Matsuo, M. Hashimoto, and A. Koike, "A study on biometric authentication based on arm sweep action with acceleration sensor," in *Proceedings of International Symposium on Intelligent Signal Processing and Communication*, 2006, pp. 219–222.

[33] J. Liuand L. Zhong, J. Wickramasuriya, and V. Vasudevan, "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, pp. 15:1–15:10.

[34] M. Bashir, G. Scharfenberg, and J. Kempf, "Person authentication by handwriting in air using a biometric smart pen device.," *BIOSIG*, pp. 219–226, 2011.

[35] L. Xia, C-C. Chen, and J. K. Aggarwal, "Human detection using depth information by kinect," in *Workshop on Human Activity Understanding from 3D Data in conjunction with CVPR (HAU3D)*, Colorado Springs, USA, 2011.

[36] C-C. Cko, M-C. Chen, T-F. Wu, S-Y. Chen, and C-C. Yeh, "Cat motor: an innovative system to detect the behavior of human computer interaction for people with upper limb impairment," in *Proceedings of the 4th international conference on Universal access in human-computer interaction: applications and services*, Berlin, Heidelberg, 2007, UAHCI'07, pp. 242–250, Springer-Verlag.

[37] D. Uebersax, J. Gall, M. V. den Bergh, and L. V. Gool, "Real-time sign language letter and word recognition from depth data," in *ICCV Workshops*, 2011, pp. 383–390.

[38] J. Garstka and G. Peters, "View-dependent 3d projection using depth-image-based head tracking," in *Proceedings of the 8th IEEE International Workshop on ProjectorCamera Systems (PROCAMS)*, 2004, pp. 52–57.

[39] J. L. Raheja, A. Chaudhary, and K. Singal, "Tracking of fingertips and centers of palm using kinect," *Computational Intelligence, Modelling and Simulation, International Conference on*, vol. 0, pp. 248–252, 2011.

[40] K. Khoshelham and S. O. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications," *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012.

[41] P. O. Kristensson, T. Nicholson, and A. Quigley, "Continuous recognition of one-handed and two-handed gestures using 3d full-body motion tracking sensors," in *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*, New York, NY, USA, 2012, IUI '12, pp. 89–92, ACM.

[42] E. Stone and M. Skubic, "Evaluation of an inexpensive depth camera for in-home gait assessment," *Journal of Ambient Intelligence and Smart Environments*.

[43] R. Plamondon and G. Lorette, "Automatic signature verification and writer identification - the state of the art," *Pattern Recognition*, vol. 22, no. 2, pp. 107–131, 1989.