A Data Driven In-Air-Handwriting Biometric Authentication System

Duo Lu, Kai Xu, Dijiang Huang
School of Computing, Informatics, and Decision Systems,
Arizona State University, Tempe, Arizona
{duolu, kaixu, dijiang.huang}@asu.edu

Abstract

The gesture-based human-computer interface requires new user authentication technique because it does not have traditional input devices like keyboard and mouse. In this paper, we propose a new finger-gesture-based authentication method, where the in-air-handwriting of each user is captured by wearable inertial sensors. Our approach is featured with the utilization of both the content and the writing convention, which are proven to be essential for the user identification problem by the experiments. A support vector machine (SVM) classifier is built based on the features extracted from the hand motion signals. To quantitatively benchmark the proposed framework, we build a prototype system with a custom data glove device. The experiment result shows our system achieve a 0.1% equal error rate (EER) on a dataset containing 200 accounts that are created by 116 users. Compared to the existing gesture-based biometric authentication systems, the proposed method delivers a significant performance improvement.

1. Introduction

With the increasing popularity of VR headset and wearable computers, gesture input interface such as Leap Motion controller [1] or data glove [6] allows the computer to capture and understand the fine movement of hands and fingers, which further enables the application of gesture biometrics as a user authentication method [15]. Compared to the simple hand gestures and body languages that we used in the daily life, the handwriting is more closely related to the cognitive process of humans [7] and individually dependent. The writing pattern of each person is stored in the motor cortex since he or she obtains the writing ability [8]. Thus, in-air-handwriting with the tip of index finger could be a better criterion for the user authentication problem since it contains more identity-related features. Moreover, a piece of in-air-handwriting can be reproduced by the writer easily in a consistent way but difficult for others to imitate.

Our objective is to authenticate a user through a human-computer-interface (HCI) that can capture the movement of in-air-handwriting, based on both the content of the writing and the way of writing. The captured movement is represented by signals of physical states of the hand, such as position, speed, or acceleration. Many existing works utilize hand-held devices to capture the movement [3, 9, 16], where the user is required to perform a hand gesture similar to writing while holding a device equipped with accelerometers for authentication purpose. Other researchers use devices such as wearable camera [12], Leap Motion controller [4] [5], and Kinect [13] to capture the in-air-handwriting from a distance. However, extracting useful information from the signal and distinguishing legitimate users from attackers is challenging. First, it is difficult for the sensors to measure the hand movement perfectly due to limited accuracy, resolution, and sampling speed. Second, pieces of the user's handwriting are not exactly the same every time even writing the same content. As a result, tolerating sensor noise and user variance would inevitably degrades the performance of the authentication system. Hence, existing works rarely reach an EER below 1% on a dataset with reasonable size. Another challenge arises from the limited understanding of in-air-handwriting, especially on what features and algorithms to use. Although similar hand movement has been studied for gesture and handwriting recognition [2, 11, 17, 14], the task of an in-air-handwriting biometric authentication system is not recognizing similar gesture patterns or characters regardless of the subjects, but distinguishing different writers based on the patterns. As a result, various ad hoc algorithms and machine learning models are employed in existing systems.

In this paper, we propose a user authentication method using the finger tip movement of a few seconds of in-air-handwriting, captured by a wearable sensor. The method is based on a simple fact that signals of the same writing content generated by the same user look similar in shape and statistics. The major contribution of this paper is threefold: First we provide a feature analysis on the in-air-handwriting movement, which helps us build classifier for separating those signals generated by the true user from those gen-

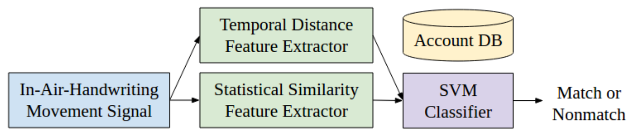
Temporal Distance Feature Extractor
Account DB
In-Air-Handwriting Movement Signal
Statistical Similarity Feature Extractor
SVM Classifier
Match or Nonmatch

Figure 1. Authentication system architecture.

erated by attackers. Second, we constructed a data driven solution for the authentication task, by training a classifier using the data at initial enrollment. This method avoids hand-craft of matching algorithm and the requirement of hand tuning of the authentication system, which makes it easy to understand and deploy. Third, we achieve a significant performance improvement compared to existing work. On a dataset collected by ourselves with 200 accounts from 116 users, our prototype system achieved 0.1% EER.

The rest of the paper is organized as follows. Section 2 explains the structure of the proposed authentication system. Section 3 contains the analysis of features we use to construct the classifier, and section 4 shows the performance results. In section 5 we discuss details of our design idea and draw the conclusion.

2. Authentication System

Our authentication system contains the following components (shown in Figure 1): a device to capture the in-air-handwriting movement, an account database, feature extractors, and an SVM classifier.

2.1. Hand Movement Capture Device

In our prototype system, we use a custom data glove as the hand movement capture device. The data glove has a microelectromechanical inertial measurement unit (IMU) with tri-axis accelerometer and gyroscope on the finger tip, and a battery powered Arduino compatible microcontroller on the wrist, shown in Figure 2 (in this paper we only use the sensor on the index finger). The inertial sensor on the glove generates data samples of acceleration and angular speed at 50 Hz. The microcontroller filters out high frequency components above 10 Hz, and computes the absolute orientation in Euler angles for each data sample and removes the gravity influence on the acceleration. The obtained in-air-handwriting signal contains the time series of data samples represented as a $l \times d$ matrix S, where l is the number of samples (usually from 100 to 500), and d is the number of sensor axes. There are 9 sensor axes used in our system, including acceleration in x, y, z, with $\pm 4g$ range, angular speed in x, y, z, with $\pm 2000 dps$ range, and Euler angles in ϕ, θ, ψ, with $\pm 180°$ range. Our authentication method is not restricted to our data glove. As long as a gesture interface can provide the physical states or trajectory of the fingertip for in-air-handwriting, this method can be applied.

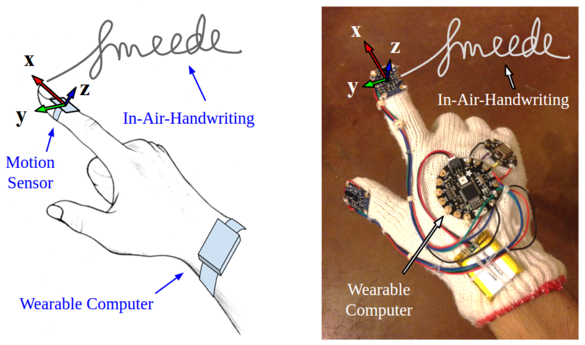

Figure 2. Hand movement capture device illustration (left) and the custom data glove prototype we made (right).

2.2. Account Database

Each account is a tuple of $< ID, template >$, where the stored passcode template is a special signal with extracted statistics. At initial enrollment, the user creates an account by providing a unique account ID and write a passcode five times for template construction. After the account creation, the user can be authenticated to login to the account by providing the account ID and writing a passcode in the air. The captured in-air-handwriting signal is compared with the stored template, and features are extracted. Then a binary SVM classifier determines whether the signal is generated from the true user based on the features, and hence, accept or reject the login request.

2.3. Feature Extractor

Once the login signal is obtained, we run the following feature extractors:

1) Temporal distance (TD), the histogram of scaled element-wise distance in temporal domain between the signal and the template, detailed in section 3.1.

2) Statistical similarity (SS), the similarity of statistics of the signal and the template, such as mean, variance, entropy, etc., detailed in section 3.2.

We define a data point $\mathbf{x}$ as a vector of combined features obtained by these two types of extractors, given a login signal and an account, *e.g.,* $\mathbf{x} = (\mathbf{TD}, \mathbf{SS})$.

2.4. Support Vector Machine Classifier

Each data point has a class label y representing whether the signal is generated by the legitimate account owner writing the correct passcode (*i.e.,* true-user class, $y = -1$) or not (*i.e.,* not-true-user class, $y = 1$). At initial enrollment, the labels are provided by the user as ground truth to train the Support Vector Machine (SVM) classifier, and at authentication stage the classifier predict the label for a newly generate data point built by the signal in the login request, so as to make a match or non-match decision. SVM is a linear binary classifier widely used in pattern recognition and other biometric authentication systems such as [18]. It seeks a hyperplane $f(\mathbf{x}) = \mathbf{wx} + b = 0$ to maximize the separation of the data points from the two classes. Those data points determining the maximum margin are called support vectors. Training an is essentially a quadratic programming problem, which can be solved efficiently.

Since all the features we use are essentially difference between the signal and the template in various aspects, if a data point is from the true-user class, most elements of the feature vector should be small, because signals obtained by the true user writing the same passcode should be almost the same in shape and statistics. Otherwise, if a data point is from the not-true-user class, only a few element can happen to be small while the majority are large. In a high dimensional space, it is possible to find a hyperplane to separate the two classes, even with only a few data points at training stage. Once the hyperplane is derived, the prediction can be made by first calculating the SVM score $f(\mathbf{x}) = \mathbf{wx} + b$ for a testing data point $\mathbf{x}$ and then comparing with a decision threshold. For example, if $f(\mathbf{x}) > threshold$, the predicted label is $y = 1$ (*i.e.,* not-true-user class). This can be interpreted that if the difference between the testing signal and the template exceeds the threshold, the testing signal is rejected. The threshold is a tunable system parameter to trade off the tolerance of the number of false matches and the number of false non-matches.

2.5. Dataset

The following three datasets are collected for evaluation:

1) 200 accounts were created by 116 users. For each of the account, we ask the user to write a passcode in the air five times as initial enrollment, take a rest, then write it five times again as testing. The users are diverse in gender (51 male, 65 female), age (from 17 to 65), education (from middle school student to university professors), occupation (including both office workers and non-office workers). There is no constraints on the content of the passcode, so the user can pick whatever they want to write as a passcode. Here we consider the authentication of each account individually, *i.e.,* one user can create multiple accounts.

2) We asked seven impostors to spoof each passcode in the first dataset for five times. The impostors are informed with the content of the spoofed passcode.

3) Additionally, seven users (the seven impostors in the second dataset) generated 17 passcodes and we asked them to write the passcodes five times every three or four days, for a period of four weeks. This dataset is mainly used for visualization due to its limited size.

For the five instances in initial enrollment, two of them are used to construct the template signal and the remaining three are used to train the SVM classifier. For the five testing signals, if we use a signal S from account A as a login signal for the account A, we create a data points of the true-user class; while if we use S as a login signal for a different account B, we create a data point of the not-true-user class (also called false-user class if it is necessary to be differentiated with spoofing data points). Data points created by spoofing signals in the dataset 2 are special instances of the not-true-user class and they are treated separately.

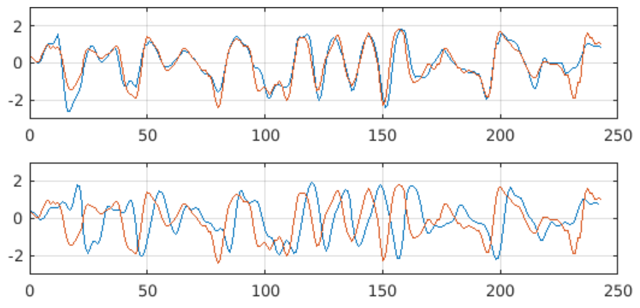
Figure 3. A pair of signals generated by a user writing the same passcode twice, after (upper) and before alignment (lower).

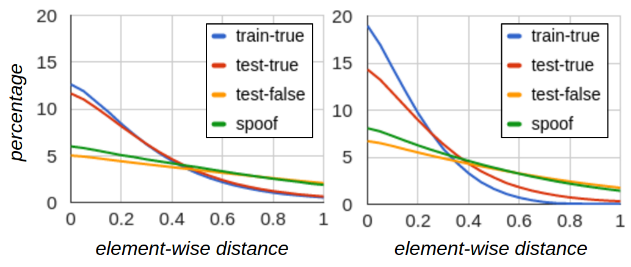

Figure 4. Example of temporal distance histogram, before scaling (left) and after scaling (right).

3. Feature Analysis

3.1. Temporal Distance

Since a testing signal should resemble the template in shape if it is from the true-user class, the signal distance in temporal domain can be used to determine whether it is generated by the legitimate user. However, when a user writes the same passcode multiple times, the captured signal varies slightly. To accommodate this variation, we use Dynamic Time Warping (DTW) to align the testing signal S against the template signal T, where the template signal is constructed at initial enrollment in the same way by aligning one signal to the other signal and taking their average. To further improve the alignment performance, we normalize each column of the signal and the template by subtracting the mean and divided by the variance of that column. Figure 3 shows the effectiveness of alignment.

After the alignment, for each sensor axis, we calculate element-wise distance $D_{ij} = |S_{ij} - T_{ij}|$. Then, we scale the element-wise distance by $k1$ and $k2$, where $k1$ is a function of template uncertainty, and $k2$ is a function of tem-

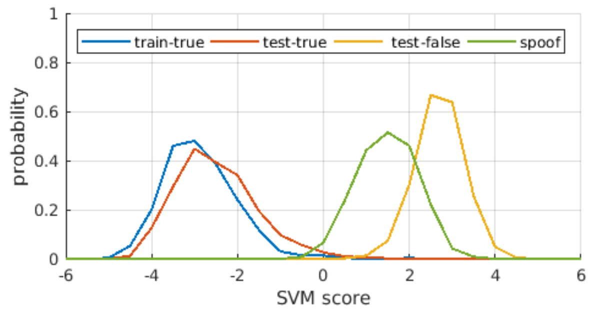

Figure 5. Score distribution for classification with TD features.

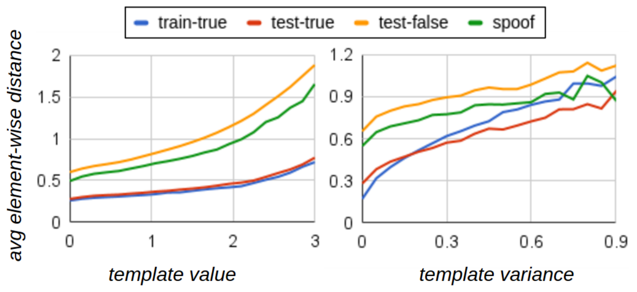

Figure 6. Change of average element-wise distance respect to template value (left) and variance (right).

plate signal value T_{ij}, detailed below. The distance features are obtained by taking the histogram of the scaled element-wise distance for each sensor axis, *e.g.,* for sensor axis j, $\mathbf{TD_j} = histogram(\{D_{ij} * k1 * k2\})$. We choose 20 bins with bin size 0.1 and range from 0 to 2 for the histogram. The final temporal distance feature vector is $\mathbf{TD} = (\mathbf{TD_1}, ..., \mathbf{TD_9})$ with 180 elements in total. An example of the histogram before and after scaling is shown in Figure 4. In this figure we show the temporal distance features normalized as the probability distribution and averaged among all the sensor axes (*i.e.,* average of $\mathbf{TD_1}$ to $\mathbf{TD_9}$). In Figure 5 we show the SVM score distribution normalized as the probability distribution by running the classifier with temporal distance features. The overlap of the distribution between two classes which needs to be separated is called the estimated Bayes error, indicating the quality of the features for distinguishing the legitimate users and the attackers. A good collection of features should derive low estimated Bayes error, *e.g.,* the temporal distance features achieve an estimated Bayes error of 0.42% without spoofing (overlap of "test-true" and "test-false" in Figure 5) and 2.9% with spoof (overlap of "test-true" and "spoof").

The objective of our distance calculation algorithm is shrinking the distance among the data points from the true-user class, while pushing away the data points with different labels, so as to obtain a better separation of these two classes. The two scaling factors can help us to achieve this goal and improve performance.

First, element-wise distance grows with template uncertainty C_{ij}, which is defined as the element-wise distance of the two signals forming the template, *i.e.,* $C_{ij} = |T1_{ij} - T2_{ij}|$. In Figure 6 we shown the change of average element-wise distance respect to template variance, and it can be observed they have a positive correlation. Without consideration of this phenomenon, some signals that actually generated by the true user might be mis-classified because of larger element-wise distances by comparing some local parts of the signal and the template. For these local parts, the element-wise distances are largely due to the inherently large variation of the movement, which can be indicated by the template uncertainty, instead of different shape. Thus, a scale factor $k_1 = 1/(1 + w_1 * C_{ij})$ is applied, which is equivalent to apply higher tolerance for these local parts of the signal. On the other hand, a data point from the not-true-user class would have large element-wise distance everywhere regardless of the scaling, because they have different shape and they can not be perfectly aligned.

Second, we also observed that element-wise distance grows with the template value, and this is independent from the previous phenomenon. In the Figure 6 we show the average element-wise distance (average of D_{ij}) respect to template value T_{ij}. We can see for signals generated by the true user, some large D_{ij} is caused by the subtraction of two large value S_{ij} and T_{ij}, not by their shape difference. Thus, a similar scale factor $k_2 = 1/(1 + w_2 * T_{ij})$ is applied independently to compensate this situation. This also explains why some researchers use nonlinear quantization and saturation of the signal to help performance [9]. In that case both the signal and the template are suppressed where their values are large. In our system, the scaling parameter w_1 and w_2 are determined by empirical results (basically choosing those values that can flatten the curve in the Figure 6 after the scaling factors are applied).

3.2. Statistical Similarity

If the signal and the template look similar in shape, they are also similar in various statistical aspects. Temporal distance does not utilize these statistics due to the process of normalization, alignment and histogram calculation. In our system we use the following five statistical features. Here d is the number of sensor axes, and l is the length of the time series.

1. The mean of each sensor axis $\mathbf{M} = (\mu_1, ..., \mu_d)$, where $\mu_j = mean(S_{1j}, ..., S_{lj})$.

2. The variance of each sensor axis $\mathbf{\Sigma} = (\sigma_1, ..., \sigma_d)$, where $\sigma_j = var(S_{1j}, ..., S_{lj})$.

3. The correlation between pairs of adjacent sensor axes $\mathbf{P} = (\alpha_{xy}, \alpha_{yz}, \alpha_{xz}, \beta_{xy}, \beta_{yz}, \beta_{xz}, \gamma_{xy}, \gamma_{yz}, \gamma_{xz})$, where α_{xy} is the correlation of acceleration axis a and

b, β_{ab} is the correlation of angular speed axis a and b, and γ_{ab} is the correlation of Euler angle axis a and b.

4. Sum of amplitude of each axis $\mathbf{\Lambda} = (\lambda_1, ..., \lambda_d)$, where $\lambda_j = \sum_{i=1}^{l} |S_{ij}|$.

5. Signal entropy of each axis $\mathbf{H} = (\eta_1, ..., \eta_d)$, where $\eta_j = -\sum_{i=1}^{l} p(S_{ij}) log_2 p(S_{ij})$. Here S_{ij} is treated as random variables and $p(S_{ij})$ is the probability density function (PDF) of S_{ij} estimated by discrete samples.

6. Length l

Collectively, the statistical feature vector of a specific signal S is the combination of these five types of statistical features, defined as $\mathbf{SF(S)} = (\mathbf{M}, \mathbf{\Sigma}, \mathbf{P}, \mathbf{\Lambda}, \mathbf{H}, l)$, and statistical similarity of signal S and template T is defined as $\mathbf{SS} = abs(\mathbf{SF(S)} - \mathbf{SF(T)})$, where $abs()$ is the element-wise absolute operation.

We can separate the signal and the template into multiple segments and apply statistical feature extraction individually on these segments, so as to create more features. This idea is based on the fact that one local segment of the signal is largely uncorrelated with another segment. For example, a user can write “PASSCODE” or “PASSPORT” as two different passcode, while the first half of the in-air-handwriting would be irrelevant to the second half. Thus, comparing each segment individually could potentially provide more information for classification, since signals generated by the true user writing the correct passcode multiple times should be consistent in statistical feature for all the segments. Given 9 sensor axes and the whole signal without segmentation, we will have 45 statistical similarity features for a data point, while segmentation of the signal and the template to 3 parts of equal length will produce 3 times in the amount of statistical features. However, a larger amount of small segments suffer from misalignment more significantly, and the curse of dimensionality makes training harder given a larger amount of features. In Figure 7 we show the classification results in estimated Bayes error using only statistical similarity features with different number of segments. In our authentication system we use 3 segments. Figure 8 shows the limited discriminating capability of individual statistical feature when used alone.

In Figure 9 we show the correlation of all the 45 statistical features of the training signals from the dataset 1, without signal segmentation. It can be seen acceleration and angular speed axes have higher positive correlation in the first three statistical features (mean, variance, and amplitude), and the entropy features of each sensor axis have a very high positive correlation with each other. These results are reasonable because if the user writes the passcode with larger movement or higher speed, acceleration (which is caused by linear force) and angular speed (which is caused by torch on

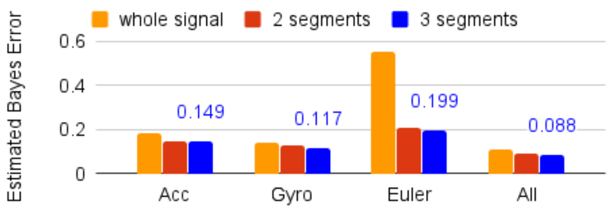

Figure 7. Classification results using statistical similarity features of different sensor axes under various number of segments.

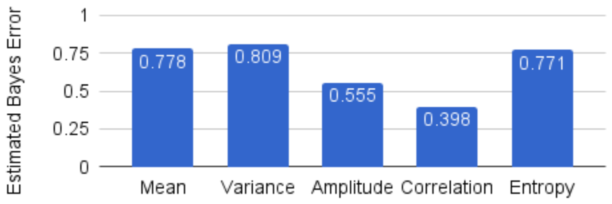

Figure 8. Classification results using different types of statistical similarity features of all sensor axes.

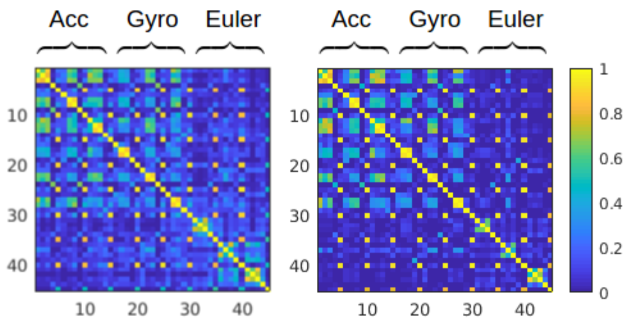

Figure 9. Visualization of statistical features of the signals, for the true-user class (left) and not-true-user class (right). Negative correlation are saturated to zero.

finger joints and wrist) would be influenced in the same direction (more intensive). Similarly if the movement itself is more complicated (more entropy), all axes of the signal would be more complicated together (high positive correlation). Since these statistical features are not all strongly chained together, they can be used jointly with temporal distance for classification.

Another useful statistical feature is the length difference between the signal and the template. If a data point is from the true-user class, the length difference is in general small (less than 20%), while if a data point is from the not-true-user class, the length difference are most likely large. It is also a feature with limited discriminating capability, which only helps when used together with other features.

In Figure 10 we show a visualization of how the data points of the two classes are clustered. The left figure shows the data points in statistical similarity feature space obtained

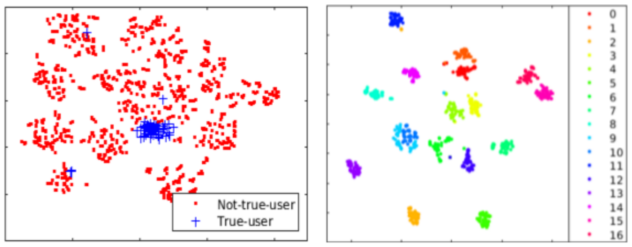

Figure 10. Visualization of data points in statistical similarity feature space (left) and raw statistical feature space (right).

from the signals of 20 accounts. The right figure shows a similar visualization in the raw statistical feature space, obtained from the 17 passcodes in dataset 3. Both figures are created by the t-SNE [10] technique to embed the high dimensional data points to 2D for visualization.

4. Classification Results

Based on the previous feature analysis, we can build a single SVM classifier for all accounts, and the classification results are shown in the first two rows of Table 1. These performance metrics are obtained by varying the decision threshold on the SVM score. During the experiments, we observed that the gap between the true-user class and the not-true-user class varies from account to account. Thus, we can also build individual SVM classifier for each account, and introduced a per account threshold offset. This per-account classifier further improves performance, shown in the third row as SVM*(TD, SS). Compared with plain DTW (the fourth row), our method has significant performance improvement. The EER is shown in Figure 11, and the ROC curve is shown in Figure 12. In these two figures, we use per-account SVM classifier with both temporal distance and statistical similarity features on three segments.

The reason for the performance improvement is threefold. First, we exploit the inherent large passcode space and the rich information in the series of strokes of hand movement through the designed features. Second, signals generated by a user writing the same passcode in our dataset are more consistent because we eliminated the constraints on the in-air-handwriting by using a wearable device instead of a stationary camera or hand-held device. This consistency is further strengthened by in the data processing stage, including removing the influence of gravity, normalization. This can be revealed by the high DTW baseline performance on our dataset. Third, the features we extracted have a high quality for distinguishing the two classes, and our careful process of the features including scaling the element-wise temporal distance and signal segmentation further improves the feature quality.

Table 1. Classification Results

Classifier	EER	EER (spoof)	FMR 10K	FMR 100K	Zero -FMR
SVM(TD)	0.2%	1.4%	1.8%	3.6%	5.1%
SVM(TD, SS)	0.2%	1.4%	1.5%	2.8%	3.9%
SVM*(TD, SS)	**0.1%**	**1.4%**	**0.5%**	**0.7%**	**1.5%**
DTW(baseline)	0.4%	4.2%	4.4%	8.4%	16.4%

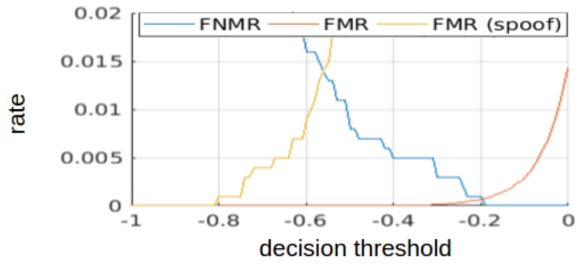

Figure 11. Equal Error Rate (EER).

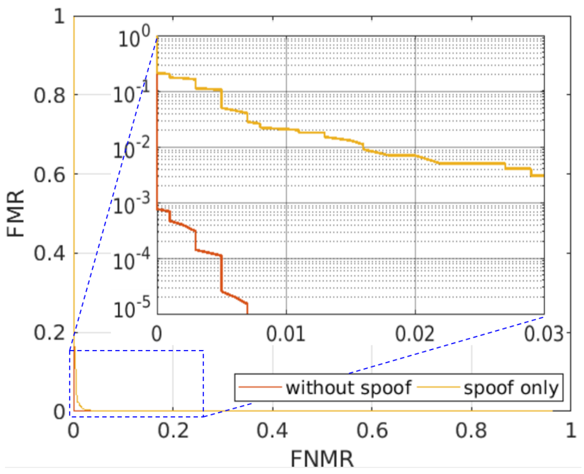

Figure 12. Receiver Operating Characteristic (ROC).

5. Conclusion and Future Work

In this paper we proposed a data driven approach of constructing in-air-handwriting biometric authentication system over gesture interface. We build a prototype system and conduct experiments on a middle scale dataset and our results shows that in-air-handwriting has good potential as a strong biometric authenticator. One possible limitation of our system is the behavior changing of the user. We conducted experiments on dataset 3 and observed small change in the SVM score with time. We believe this can be accommodated by template update techniques. Thus, in the future work we will analyze the behavior persistence and the long term effect on our system with more data collected by tracking the same set of users with a few weeks.

References

[1] Leap motion controller. URl: https://www.leapmotion.com.

[2] C. Amma, M. Georgi, and T. Schultz. Airwriting: a wearable handwriting recognition system. *Personal and ubiquitous computing*, 18(1):191–203, 2014.

[3] G. Bailador, C. Sanchez-Avila, J. Guerra-Casanova, and A. de Santos Sierra. Analysis of pattern recognition techniques for in-air signature biometrics. *Pattern Recognition*, 44(10):2468–2478, 2011.

[4] A. Chahar, S. Yadav, I. Nigam, R. Singh, and M. Vatsa. A leap password based verification system. In *Biometrics Theory, Applications and Systems (BTAS), 2015 IEEE 7th International Conference on*, pages 1–6. IEEE, 2015.

[5] A. Chan, T. Halevi, and N. Memon. Leap motion controller for authentication via hand geometry and gestures. In *International Conference on Human Aspects of Information Security, Privacy, and Trust*, pages 13–22. Springer, 2015.

[6] L. Dipietro, A. M. Sabatini, and P. Dario. A survey of glove-based systems and their applications. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(4):461–482, 2008.

[7] Y. Itaguchi, C. Yamada, and K. Fukuzawa. Writing in the air: Contributions of finger movement to cognitive processing. *PloS one*, 10(6):e0128419, 2015.

[8] S. R. Klemmer, B. Hartmann, and L. Takayama. How bodies matter: five themes for interaction design. In *Proceedings of the 6th conference on Designing Interactive systems*, pages 140–149. ACM, 2006.

[9] J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan. uwave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657–675, 2009.

[10] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[11] C. Qu, D. Zhang, and J. Tian. Online kinect handwritten digit recognition based on dynamic time warping and support vector machine. *Journal of Information and Computational Science*, 12(1):413–422, 2015.

[12] H. Sajid and S. C. Sen-ching. Vsig: Hand-gestured signature recognition and authentication with wearable camera. In *Information Forensics and Security (WIFS), 2015 IEEE International Workshop on*, pages 1–6. IEEE, 2015.

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

[14] H. Wen, J. Ramos Rojas, and A. K. Dey. Serendipity: Finger gesture recognition using an off-the-shelf smartwatch. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 3847–3851. ACM, 2016.

[15] J. Wu. *Gesture passwords: Concepts, methods, and challenges*. PhD thesis, Carnegie Mellon University, 2016.

[16] A. Zaharis, A. Martini, P. Kikiras, and G. Stamoulis. User authentication method and implementation using a three-axis accelerometer. In *International Conference on Mobile Lightweight Wireless Systems*, pages 192–202. Springer, 2010.

[17] Y. Zhang and C. Harrison. Tomo: Wearable, low-cost electrical impedance tomography for hand gesture recognition. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pages 167–173. ACM, 2015.

[18] 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*, pages 139–150. ACM, 2011.