Trajectory based Behavior Analysis for User Verification*

Hsing-Kuo Pao[1], Hong-Yi Lin[1], and Kuan-Ta Chen[2]

[1] Dept. of Computer Science & Information Engineering,
National Taiwan University of Science & Technology, Taipei, Taiwan
{pao, M9615061}@mail.ntust.edu.tw
[2] Institute of Information Science, Academia Sinica, Taipei 115, Taiwan.
ktchen@iis.sinica.edu.tw

Abstract. Many of our activities on computer need a verification step for authorized access. The goal of verification is to tell apart the true account owner from intruders. We propose a general approach for user verification based on user trajectory inputs. The approach is labor-free for users and is likely to avoid the possible copy or simulation from other non-authorized users or even automatic programs like bots. Our study focuses on finding the hidden pattern on the trajectories produced by account users. We employ a Markov chain model with Gaussian distribution in its transitions to describe the behavior on the trajectory. To distinguish between two trajectories, we propose a novel dissimilarity measure combined with a manifold learnt tuning for catching the pairwise relationship. Based on the pairwise relationship, we plug-in any effective classification or clustering methods for the detection of unauthorized access. The result shows that the proposed method can accurately verify the user identity. The similar method may also be applied for the task of recognition, predicting the trajectory type without pre-defined identity.

***Keywords*:** *Verification, Behavior analysis, Account security, Trajectory, Dissimilarity measure, Manifold learning, Isomap.*

1 Introduction

With the network grows fast, people rely on Internet for daily activities. Someone uses the Internet to do bank transactions, talk with friends in the electronic community, search interesting information, play on-line games, and so on. Many of the activities do not allow anonymous access. The usual login-on process is to provide password or even biometrics like fingerprint, face, iris for personal identification. In this case, the users from both sides of network face the problem that the personal information may be robbed by some unauthorized person such as the cracker breaking-in others' on-line game accounts for illegal benefits. Consequently, *verification* plays an important role in *account security*. In this work,

* Research partially supported by Taiwan National Science Council Grant # 98-2221-E-011-105.

we propose a verification scheme based on *user trajectories* to check whether or not the person is the true account owner.

Two-factor authorization is a mechanism that is used to authorize or verify the ownership of an account. Having two factors at the same time to verify identity, the account security is enhanced. For an example, the first factor is that users have to slide the smart card into ATM, and the second factor is for users to type their passwords. Nevertheless, those factors can possibly be faked or stolen by hackers. We suggest to analyze the behavior of users' trajectories so that different patterns from two users can be recognized. Although hackers can design some illegal programs which learn decisions of humans, the behavior patterns are generally difficult to simulate because it is an AI-hard problem. In addition, based on the proposed method, we try not to bring users any extra load to assist the authorization methods to improve the authorization power. That is to say, the users do not need to adjust their ways of account usages. We define our problem as follows:

Definition 1 (Verification) *Given a coordinate trace, Verification is to match between a trace and a pre-defined identity, a Yes/No question.*

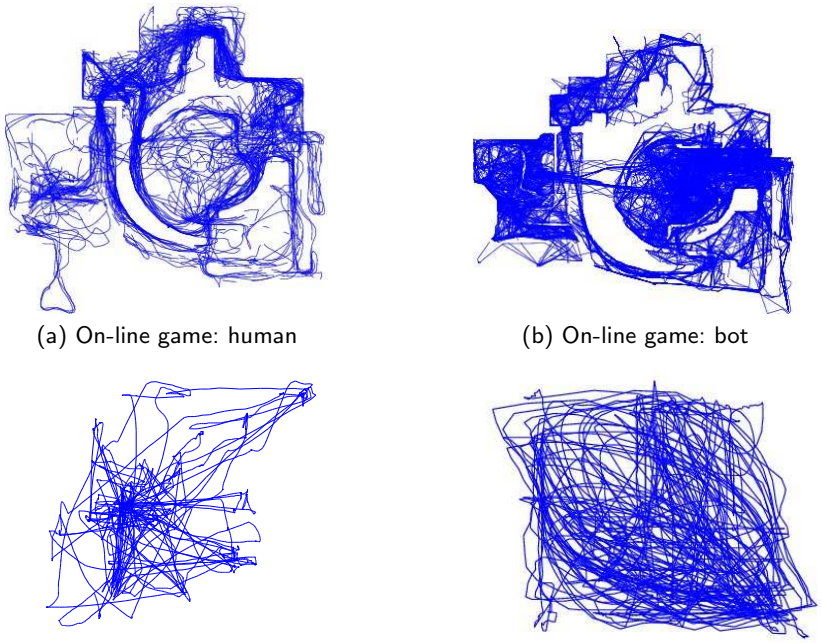
Proposed Method As shown in Figure 1, Given an input of coordinate set, we want to extract hidden patterns. We consider a trace $\mathbf{s} = (\mathbf{x}_1, \ldots, \mathbf{x}_t, ..., \mathbf{x}_T)$ which has T coordinates in a 2-D or 3-D space where T is the length of the trace. Working on a trace of short length implies that the method is effective even in such a limited-length input. Given the coordinate information, we want to extract features from the trace and compute a dissimilarity measure for each pair of traces. Afterwards, we combine the pairwise dissimilarities with a manifold learning approach called *Isomap* (*Isometric feature mapping* [1]) for trajectory representation and use *Smooth SVM* [2] in the representation space to classify the patterns into the true user or an intruder. Various trajectory data will be investigated by our method.

2 Related Work

In this section, we discuss some issues which are relevant to our research. There are two topics: account verification and trajectory analysis.

Verification We need to restrict the account access only for the true account owner. To identify who the user is or at least verify whether the user is the account owner, we discuss several approaches.

Signature Signature verification is a well known method to verify whether a user is the account owner. The signature verification can be roughly divided into two categories: on-line signature verification [3] and off-line signature verification [4]. In on-line verification, the input is the handwriting trajectory, measured by time.



(a) On-line game: human (b) On-line game: bot

(c) Mouse trace: a left-handed user (d) Mouse trace: a right-handed user

Fig. 1. The different kinds of trajectory input. The (a) and (b) are the traces of bot and human respectively from an on-line game called Quake 2. The (c) and (d) are mouse traces from two individuals. We can vaguely observe that (c) is from a left-handed user and (d) is from a right-handed user.

Mario et al. [5] used a camera to collect such handwriting traces. Richiardi et al. [6] employed the GMMs (*Gaussian Mixture Models*) to verify on-line signatures. They used Gaussian components to represent the features of handwriting, and used MDL (*Minimum Description Length*) principle to automatically select the signature model. In off-line signature, the input is usually a 2-D signature image captured by scanner. The problem therefore becomes an image recognition problem. Usually, we believe that the on-line version rather than the off-line version extracts more information because the time stamps are included in the input.

CAPTCHA For bot detection, the CAPTCHA (*Completely Automated Public Test to tell Computers and Humans Apart*) [7] is a test that automatically asks users some problems to judge if the user is a human or a bot (automated program). The problem may be a randomly generated string drawn on an image. That is easy for a human, but may be difficult for a bot to tell what the string

is. The method is effective, even is still able to be cracked; however, answering the trivial questions can be annoying for human users.

Trajectory Analysis The pattern recognition for sequential or trajectory data is a wide area of research. We discuss only the ones most related to our work. There are mainly two types of sequential data. One holds temporal relation such as handwriting traces, mouse traces, or avatar traces from online games. The other type may not hold the temporal relation, for instance, biological sequences or language texts. For handling traces, the SAX (*Symbolic Aggregate approXimation*) [8] is a popular method. The method is successfully applied to many applications. One of the key steps of SAX is to discretize the numerical values of input to produce a set of symbols which is an approximation of the original input. Jae-Gil et al. [9] combine the region-based and trajectory-based clustering methods to classify trajectories. They use MDL principle to partition trajectories, and then search for specific patterns. Keogh et al. [10] studied parameter-free description of sequential data. Pao et al. [11] considered the distance function between biological sequences. Both followed the study of Li et al. [12], who tried to use Kolmogorov complexity [13] to describe the "irregularities" in sequential data. The Kolmogorov complexity of a finite sequence is generally incomputable. However, some compression methods [11] can be adopted to obtain its approximation.

3 Framework

In this section, we introduce our framework and discuss how we deal with the verification problem based on trajectories. The organization of our framework is as follows. The first step is to extract useful features from trajectories. The second step measures dissimilarities between a pair of sequenes. Based on the dissimilarity measure, the last step is to refine the dissimilarities by a manifold learning method called Isomap [1] and use a classification method to classify trajectories into normal users or intruders. We adopt Smooth SVM [2] as the classifier.

3.1 Feature Extraction

Given a trajectory $\mathbf{s} = (\mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_T)$ of T seconds, we extract useful features from the trajectory. There are two different features in our study. First, a step is a vector $\mathbf{x}_{t+1} - \mathbf{x}_t$, so Euclidean step size is $\lambda = ||\mathbf{x}_{t+1} - \mathbf{x}_t||$. We define the step size change to be $\Delta\lambda_t = \lambda_{t+1} - \lambda_t$. Second, an angle θ_t is defined to be the angle between the vector $\mathbf{x}_{t+1} - \mathbf{x}_t$ and the x-axis. The information of angle change is $\Delta\theta_t = \theta_{t+1} - \theta_t$.

3.2 Dissimilarity Measures

We employ the Markov chain model for dissimilarity measurement. Let $\mathcal{M}(\sigma_\lambda, \sigma_\theta)$ denote the associated model of a trajectory sequence (probably based on the

maximum likelihood principle), where σ_λ and σ_θ are the transition parameters. The σ_λ describes the standard deviation of step size λ_{t+1}, assumed centered in λ_t; and the σ_θ describes the standard deviation of angle θ_{t+1}, assumed centered in θ_t. That is, based on the Markovian properties, between two coordinates in consecutive time stamps $\mathbf{x}_t$, $\mathbf{x}_{t+1}$, we assume that

$$P(\lambda_{t+1}|\lambda_t) \sim N(\lambda_t, \sigma_\lambda^2) = \frac{1}{\sqrt{2\pi}\sigma_\lambda} \exp\Big(- \frac{(\lambda_{t+1} - \lambda_t)^2}{2\sigma_\lambda^2}\Big)\,, \tag{1}$$

$$P(\theta_{t+1}|\theta_t) \sim N(\theta_t, \sigma_\theta^2) = \frac{1}{\sqrt{2\pi}\sigma_\theta} \exp\Big(- \frac{(\theta_{t+1} - \theta_t)^2}{2\sigma_\theta^2}\Big)\,. \tag{2}$$

Given a model $\mathcal{M}$, the log-likelihood $\ell(\mathbf{s};\mathcal{M})$ of a trajectory $\mathbf{s}$ can be written as

$$\ell(\mathbf{s};\mathcal{M}) = \log L(\mathbf{s};\mathcal{M}) = \log P(\mathbf{x}_1) + \sum_{t=1} \log\Big(P(\mathbf{x}_{t+1}|\mathbf{x}_t)\Big)\,, \tag{3}$$

where L is the likelihood function. In our design, the dissimilarity or distance between two trajectories depends on how well a trajectory is described by the model for the other trajectory. First, given the model $\mathcal{M}$ we compute the code length of a trajectory $\mathbf{s}$ as a negative logarithm of the likelihood, such as

$$c(\mathbf{s}|\mathcal{M}) = -\ell(\mathbf{s};\mathcal{M}) = -\log L(\mathbf{s};\mathcal{M})\,. \tag{4}$$

Note that $\mathcal{M}$ does not have to be the associated model of the trajectory $\mathbf{s}$. We define the dissimilarity between two trajectories $\mathbf{s}_1$ and $\mathbf{s}_2$ as

$$d(\mathbf{s}_1, \mathbf{s}_2) = \frac{c(\mathbf{s}_1|\mathcal{M}_2) + c(\mathbf{s}_2|\mathcal{M}_1)}{c(\mathbf{s}_{12}|\mathcal{M}_{12})}\,, \tag{5}$$

where $\mathbf{s}_{12}$ is the trace concatenating $\mathbf{s}_1$ and $\mathbf{s}_2$ one after another, and $\mathcal{M}_{12}$ is the associated model of $\mathbf{s}_{12}$.

3.3 Trajectory Representation and Labeling

In principle, given the pairwise dissimilarities for trajectories as in Eq. 5, we can simply adopt a simple method such as k nearest neighbors to classify trajectories to be one belonging to the true account owner or one belonging to an intruder. However, we aim at designing a more effective method. We seek an embedding feature space to represent a set of trajectories/sequences; and in the space, we adopt an SVM classifier to label the sequences. In the feature space, two sequences are close (similar) to each other if (1) they have small measure of Eq. 5; or, (2) both of them are close (similar) to a third sequence. The second condition implies that two sequences $\mathbf{s}_1$ and $\mathbf{s}_2$ are similar, if they are both similar to some other sequence. In order to achieve the goal, we apply Isomap [1] to find the feature space, that is, the representation of the trajectories. In Isomap, we

1. construct a neighborhood graph by linking each pair of points that qualify as neighbors;

2. find the length of the shortest path between each pair of points and take it as the approximation of their geodesic distance; and
3. take the pairwise (geodesic) distances as the input and apply Multidimensional Scaling (or MDS) [14] to find the global Euclidean coordinates of the points.

Figure 2 shows an example of the 2-D plot after the process of Isomap. The "optimal" dimensionality (called *intrinsic dimensionality*) for separating different kinds of trajectories can be estimated by finding the "elbow" point in the residual variance curve [1]. In the feature space, ideally, we can adopt any classifier to verify a trajectory to be one from the true account owner or one from others. In this study, we use SSVM [2] to evaluate the performance of proposed method.

4 Experiment

Data Description Our experiment datasets are real data including handwriting, mouse traces, and the traces from on-line games. The handwriting dataset (UJI Pen Characters[3]) collects handwriting of digits, lowercase and uppercase letters from 11 humans. We created the mouse movement dataset based on eight users' daily mouse controlling traces, a total of 178 instances. The game trace data is to collect data from a popular online game called Quake 2, a famous FPS (First-Person Shooter) game developed by id Software[4], a trace data lasting about 143.8 hours in total. The dataset consists of some traces from human users, and some others from well-known game bots (automatic programs) such as CR Bot[5], Eraser Bot[6] and ICE Bot[7]. In summary, the data statistics as well as the parameters used in this work are listed in Table 1.

Table 1. Data statistics and parameters. The k_{Iso} is the k used in constructing the neighborhood graph for Isomap. For simplicity, the intrinsic dimensionality is chosen to be 5 in all datasets.

Name	Users	Instances	Trace Length	k_{Iso}	Intrinsic Dim.
Handwriting	11	110	702	7	5
Mouse	8	178	16665	6	5
Game	94	940	1000	8	5

In this series of experiments, we want to verify if a trajectory belongs to the true account owner. We take turns to investigate trajectories from all pairwise individuals. In other words, if there are n users in the dataset, we do C_2^n

[3] http://archive.ics.uci.edu/ml/datasets/UJI+Pen+Characters
[4] http://www.idsoftware.com/
[5] Version 1.14, http://arton.cunst.net/quake/crbot/
[6] Version 1.01, http://downloads.gamezone.com/demos/d9862.htm
[7] Version 1.0, http://ice.planetquake.gamespy.com/

Table 2. (a) The summary of the verification results on various inputs, and (b) the recognition result on the Game dataset, with different lengths of inputs. The table shows the average error rates, in percentage by SSVM classification, with three times of ten-fold cross-validation. In (b), the dataset includes four classes: human, and three types of bots: CR Bot, Eraser Bot, and ICE Bot.

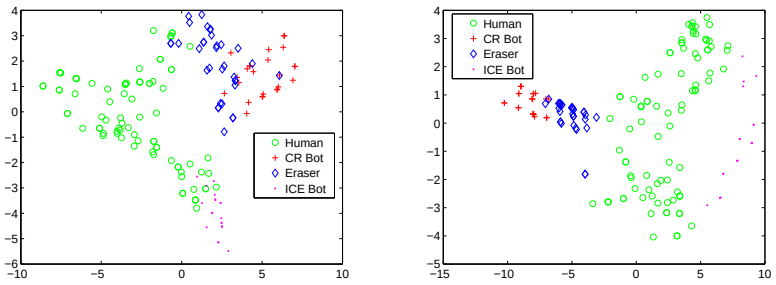
Data Set	Training Error	Test Error
Handwriting	1.47	2.89
Mouse	6.82	8.79
Game	7.59	14.34

(a)

Trace Length	Training Error	Test Error
500 seconds	5.48	7.97
1000 seconds	2.15	2.83

(b)

tests. Table 2(a) shows the performance of proposed method in different kinds of trajectories. Among them, the handwriting dataset gives us the best accuracy (97.11%), followed by the accuracy for mouse trace verification (91.21%), then the accuracy of game traces (85.66%). It follows our intuition. The handwriting traces give us better discriminative power, than mouse traces. On the other hand, game traces are usually in a restricted environment, therefore lack some degree of freedom to show the true identity of trace users.



(a) Quake 2: 500 seconds (b) Quake 2: 1000 seconds

Fig. 2. Given different length of input, the representation of the Quake 2 traces after the projection by Isomap into a 2-D space, where a point represents a trace of a human user (green circle) or from a bot (others). The x- and y-axes are the first and second principal coordinates from Isomap. Classification is worked on a higher dimensional space, known as the space of *intrinsic dimensionality*, shown in Table 1.

To further demonstrate the power of proposed method, we use our method to extract different patterns from human traces and three types of bot traces, a multi-class classification problem. The result is shown in Table 2(b). When the trace is 500-second long, our method can reach 92.03% accuracy. If we continue to collect the data up to 1000-second long, the accuracy can further be enhanced

to 97.17%. In Figure 2, we can visualize that the trace has a better presentation when a longer trace is collected. Obviously, we prefer a method that can identify the user behavior as fast as possible, before significant amount of account information or account value got stolen by crackers.

5 Conclusion

In this work, we proposed a novel method for user trajectory verification. The verification scheme is based on a dissimilarity measure, followed by a manifold learning adjustment from Isomap. We have applied our method to various types of trajectories including handwriting, mouse traces, and game traces. The results show that our method is effective in picking the hidden pattern in the trajectory and is plausible to solve related problems in a wide range. Thus, we believe that the proposed method merits further investigation by the account verification and related research communities.

References

1. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science **290** (2000) 2319–2323
2. Lee, Y.J., Mangasarian, O.L.: SSVM: A smooth support vector machine for classification. Comput. Optim. Appl. **20** (2001) 5–22
3. Jain, A.K., Griess, F.D., Connell, S.D.: On-line signature verification,. Pattern Recognition **35** (2002) 2963–2972
4. Qiao, Y., Liu, J., Tang, X.: Offline signature verification using online handwriting registration. In: CVPR. (2007)
5. Munich, M.E., Perona, P.: Visual identification by signature tracking. IEEE Trans. Pattern Anal. Mach. Intell. **25** (2003) 200–217
6. Richiardi, J., Drygajlo, A.: Gaussian mixture models for on-line signature verification. In: WBMA '03: Proceedings of the 2003 ACM SIGMM workshop on Biometrics methods and applications, New York, NY, USA, ACM (2003) 115–122
7. von Ahn, L., Blum, M., Hopper, N.J., Langford, J.: CAPTCHA: Using hard AI problems for security. In: EUROCRYPT. (2003) 294–311
8. Lin, J., Keogh, E.J., Lonardi, S., chi Chiu, B.Y.: A symbolic representation of time series, with implications for streaming algorithms. In: DMKD. (2003) 2–11
9. Lee, J.G., Han, J., Li, X., Gonzalez, H.: *TraClass*: trajectory classification using hierarchical region-based and trajectory-based clustering. PVLDB **1** (2008) 1081–1094
10. Keogh, E., Lonardi, S., Ratanamahatana, C.A.: Towards parameter-free data mining. In: KDD '04: Proceedings of the tenth ACM SIGKDD inter. conf. on Knowledge discovery and data mining, New York, NY, USA, ACM (2004) 206–215
11. Pao, H.K., Case, J.: Computing entropy for ortholog detection. In: International Conference on Computational Intelligence. (2004) 89–92
12. Li, M., Badger, J.H., Chen, X., Kwong, S., Kearney, P., Zhang, H.: An information-based sequence distance and its application to whole mitochondrial genome phylogeny. Bioinformatics **17** (2001) 149–154
13. Li, M., Vitányi, P.: An Introduction to Kolmogorov Complexity and Its Applications (2nd Ed.). Springer, New York (1997)

14. Cox, T.F., Cox, M.A.A.: Multidimensional Scaling, Second Edition. Chapman & Hall/CRC (2000)