# Trajectory Analysis for User Verification and Recognition[☆]

Hsing-Kuo Pao[a,*], Junaidillah Fadlil[a], Hong-Yi Lin[a], Kuan-Ta Chen[b]

[a] *Dept. of Computer Science & Information Engineering, National Taiwan University of Science & Technology, Taipei 106, Taiwan*
[b] *Institute of Information Science, Academia Sinica, Taipei 115, Taiwan.*

**Abstract**

For many computer activities, user verification is necessary before the system will authorize access. The objective of verification is to separate genuine account owners from intruders or miscreants. In this paper, we propose a general user verification approach based on user trajectories. A trajectory consists of a sequence of coordinated inputs. We study several kinds of trajectories, including on-line game traces, mouse traces, handwritten characters, and traces of the movements of animals in their natural environments. The proposed approach, which does not require any extra action by account users, is designed to prevent the possible copying or duplication of information by unauthorized users or automatic programs, such as bots. Specifically, the approach focuses on finding the hidden patterns embedded in the trajectories produced by account users. We utilize a Markov chain model with a Gaussian

[☆]Research partially supported by Taiwan National Science Council Grants # 100-2218-E-011-010 and # 100-2628-E-001-002-MY3.

[*]Corresponding author

*Email addresses:* pao@mail.ntust.edu.tw (Hsing-Kuo Pao), nedijf@gmail.com (Junaidillah Fadlil), M9615061@mail.ntust.edu.tw (Hong-Yi Lin), ktchen@iis.sinica.edu.tw (Kuan-Ta Chen)

distribution in its transition to describe trajectory behavior. To distinguish between two trajectories, we introduce a novel dissimilarity measure combined with a manifold learned tuning technique to capture the pairwise relationship between the two trajectories. Based on that pairwise relationship, we plug-in effective classification or clustering methods to detect attempts to gain unauthorized access. The method can also be applied to the task of recognition, and used to predict the type of trajectory without the user's pre-defined identity. Our experiment results demonstrate that, the proposed method can perform better, or is competitive to existing state-of-the-art approaches, for both of the verification and recognition tasks.
*Key words:*
account security, bot detection, dissimilarity measure, Isomap, manifold learning, on-line game, trajectory, verification.

---

## 1. Introduction

With the rapid growth of computer networks, an increasing number of people are relying on the Internet to perform various activities in their daily lives; for example, conducting bank transactions, talking with friends in an on-line community, searching the Web for information, or playing on-line games. Many of the activities do not allow anonymous access. To log-on to a system, the user normally provides a password, but biometric methods like fingerprint matching, facial recognition, or iris scans may also be used for personal identification. Sometimes, a web connection is built on an untrusted network and a user's personal information, and even his identity, may be stolen by unauthorized persons, such as hackers breaking into on-line game

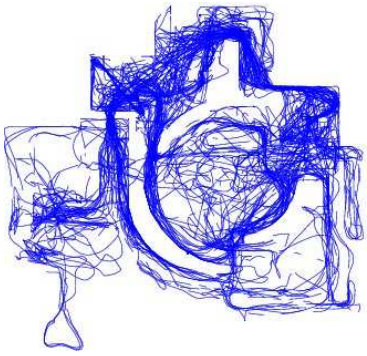
accounts. *Verification* schemes play an important role in preventing such activity. They are usually integrated with other intrusion detection methods to provide a complete defense, known as *account security*.

In this work, we propose a verification scheme based on *user trajectories*, which are sequences of coordinated inputs. The objective is to verify that the person accessing an account is the actual owner. The method can also be used to recognize the person represented by the trajectory if his identify is not provided. We evaluate our method on several kinds of trajectories, including on-line game traces, mouse traces, handwriting traces, and traces of the movements of animals in their natural environments. Some examples are shown in Figure 1. The experiment results demonstrate the method's efficacy. We formally define our problem as follows:

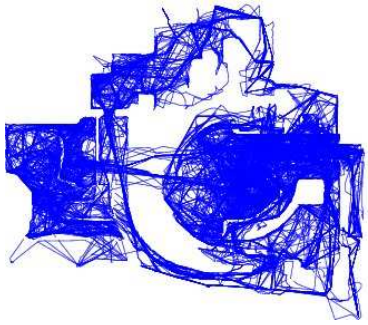
**Definition 1 (Verification and Recognition).** *Given a trajectory of coordinates* $\mathbf{s} = (\mathbf{x}_1, \ldots, \mathbf{x}_t, ..., \mathbf{x}_T)$*, where* $T$ *is the length of the trajectory and* $\mathbf{x}_t \in \mathbb{R}^2$ *or* $\mathbb{R}^3$*, verification evaluates a function* $v(\mathbf{s}, I) = y \in \{0, 1\}$ *that determines whether a match exists between a trajectory* $\mathbf{s}$ *and a pre-defined identity* $I \in \mathcal{I}$*. This is a simple Yes/No question. On the other hand, recognition evaluates a function* $r(\mathbf{s}) = I \in \mathcal{I}$ *that identifies the owner of the trajectory. This is a multiple choice question.*

Our first goal is to extract hidden patterns from a set of coordinates. We consider a trajectory of length $T$ denoted[1] by $\mathbf{s} = (\mathbf{x}_1, \ldots, \mathbf{x}_t, ..., \mathbf{x}_T)$. Sometimes we prefer a method that remains robust, even when the length

[1]We use equally-spaced time stamps in this work. In practice, we sample one point for each second in this work.

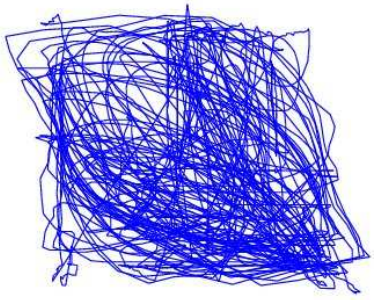


(a) On-line game: human



(b) On-line game: bot

(c) Mouse trace: a left-handed user



(d) Mouse trace: a right-handed user

Figure 1: Different types of input trajectory: (a) and (b) are the traces of a human and a bot taken from an on-line game called Quake 2; (c) and (d) are the mouse traces of two users. From their appearance, we can determine that (c) belongs to a left-handed user and (d) belongs to a right-handed user.

of the input is limited. To have that, we can detect non-authorized access in an early moment. After extracting the hidden patterns, we compute a dissimilarity measurement for each pair of trajectories, and use some well-known classifiers to decide the label of each trajectory. More specifically, the dissimilarity measurement is integrated with a manifold learning method called *Isomap* (*Isometric feature mapping*) [29] for trajectory representation.

Then, the trajectory's label is decided by a type of support vector machines, called *Smooth SVM* [13] in the representation space. A positive label means that an intruder is trying to access the account, and a negative label indicates that the person is the actual account holder. The proposed method can be applied to various types of trajectories, e.g., on-line game traces, handwriting traces, mouse traces, and traces of animal movements.

Note that our method can be regarded as a *two-factor* authorization process [27]. The ideal design would be to combine a password and our method to ensure complete account security. Our work differs from other approaches in that it is *passive*, which means that no extra effort is required by account users. For instance, user trajectories can be captured easily by a background TSR (Terminate and Stay Resident) process on a PC. Therefore, it is preferable to other two-factor schemes, such as those that combine a password with some biometric features, physical devices like smart cards, or a CAPTCHA test [30], which we discuss in the next section.

We also study the scenario where we do not have a pre-assigned identity, which is a *recognition* task. For instance, if the user does not input a password, our method can still recognize his identity by analyzing his mouse trace of a certain length. The success of the recognition process depends on the size of the account database. We assess the performance of the proposed method on datasets that only have a limited number of account users. For both of the verification and recognition tasks, we claim that the proposed method is superior to or competitive to state-of-the-art approaches that we have known.

The remainder of this paper is organized as follows. Section 2 contains

a review of related works. In Section 3, we describe our user verification and recognition method; and in Section 4, we evaluate its performance on different input traces. Then, in Section 5, we summarize our conclusions.

## 2. Related Work

In the first part of this section, we discuss previous user verification and recognition research. Because the input of this work is trajectory data, in the second part of the section, we discuss some past approaches that are relevant to trajectory analysis.

### *2.1. Account Security*

#### *2.1.1. Traditional Approach*

To prevent unauthorized access and the theft of account information, various techniques are used to verify the user's identity and match it with the account owner's pre-stored profile. Most people use a password or a PIN to access their accounts. To enhance the security level, some people maintain different passwords for different accounts, and change passwords frequently. Security can also be improved by using a password in conjunction with a communication lock. As well as providing the password, the user must unlock the communication lock by dialing a specific number, after which he has a limited timeframe to log-on to his account. Most of the above methods are inconvenient and inefficient. A user must remember his password to access his account. On the other hand, a hacker can easily compromise the account by stealing the access from an intercepted log-in process. If the system could verify the identity of account users based on their account usage patterns, such problems may be solved.

*2.1.2. Handwritten Signature and Other Biometrics*

In contrast to the above methods, which are based on "something you know" or "something you have" biometric approaches, it is believed that a method that shows "something you are" is more resistant to duplication. For example, in an on-line game, a player can steal other players' accounts and obtain the same user privileges. However, imitating a user's game-playing trajectory would be more difficult [19] because the behavior of human players is not always rational or logical.

Among all the interesting user verification and recognition methods, we focus on the handwritten signature, a well-known biometric. The handwritten signature has long been used to judge user's identity in the history. In this work, we regard the handwritten signature as a special kind of user trajectory, if we consider the temporal information in the signature. Signature verification methods can be divided into *on-line* methods and *off-line* methods [10, 25]. On-line methods consider the temporal information in a signature, but off-line methods only consider 2-D images captured by a scanner as the input. Because on-line methods consider temporal information, they are 5-25% more accurate than off-line methods [25]. However, few businesses have devices to capture on-line signatures. Moreover, it is not easy to acquire the on-line information in every case; for instance, it is difficult to obtain the on-line information of signatures on personal checks.

To acquire handwriting traces for on-line verification, Munich et al. [21] used a camera to collect data and claimed that affine arc-length parameterization methods perform better than conventional time and Euclidean arc-length methods. Richiardi et al. [26] employed *Gaussian Mixture Models* to verify

on-line signatures. They used Gaussian components to represent the features of signatures, and applied the MDL (*Minimum Description Length*) principle to automatically select the signature model. Ahmad et al. [1] combined HMM and SVM for online handwriting recognition. In off-line signature verification, the problem is similar to an image recognition problem. To bridge the performance gap between on-line and off-line methods, Qiao et al. [25] developed a method that finds on-line information from off-line inputs based on a stored on-line signature database. In contrast to the above methods, our model utilizes a very simple feature set. Even so, its performance is superior or comparable to that of existing methods[2]. Moreover, our primary motivation is to ensure account security. In this case, the devices needed to capture on-line information are available to us. For example, a mouse or an electronic pen can help us capture mouse traces, handwriting traces, and online game traces easily. Fingerprint, iris, retina, facial, and motion patterns are also used in biometric verification methods [9]. Those methods have high statistical inter-variance for identity distinguishment. Our method, which is based on behavior traits, may not have such high inter-variance between individuals; however, it does not need many extra devices[3] to acquire the input [5].

### *2.1.3. CAPTCHA*

CAPTCHA (*Completely Automated Public Test to tell Computers and Humans Apart*) [30] is a test that *automatically* asks a user to solve some

[2]It will be discussed in Section 4.

[3]Motion pattern detection usually needs some devices. For instance, we need devices to catch the emotional states from facial and voice patterns in Fujita et al. [5].

problems in order to determine if the user is a human or a bot (automated program). Although the method is effective, it can still be hacked by relaying the puzzles to human operators to obtain the correct answers. One common CAPTCHA test shows some randomly distorted words or words in a cluttered background. Answering the questions can be annoying for human users [8]. Because of the success of recently developed computer vision techniques [20], more difficult tests are being developed, and the tests are no longer trivial for human being as before.

### *2.2. Trajectory Analysis*

A great deal of research has been conducted on pattern recognition in sequential or trajectory data. We only discuss approaches that are closely related to our work. There are two types of sequential data: (1) dynamic data, such as handwriting traces, mouse traces, and avatar traces from online games; and (2) static data, such as biological sequences or language texts. Both types of data may have dependency between neighboring data elements; however, the latter usually has only limited length. We focus on dynamic data in this paper.

SAX (*Symbolic Aggregate approXimation*) [17] is widely used to process sequential data. One of the key steps of SAX involves discretizing the numerical values of the input data to produce a set of symbols that approximate the original input. Jae-Gil et al. [12] combined the region-based and trajectory-based clustering methods to classify trajectories. They used the MDL principle to partition trajectories, and then searched for specific patterns. Keogh et al. [11] suggested a parameter-free description of sequential data; while Pao et al. [22] considered the distance function between biological

sequences. Both studies followed the work of Li et al. [14], who tried to use the Kolmogorov complexity [15] to describe the "irregularities" in sequential data. Although the Kolmogorov complexity of a finite sequence is generally incomputable, some compression methods (e.g. [22]) can be exploited to obtain its approximation. In addition to the above works, Chen et al. [2, 3] and Pao et al. [23] proposed using trajectory inputs for game bot detection, and applied the method to bot detection in an FPS (First-Person Shooter) game called Quake 2. Some studies have been extended to other types of trajectories[4] in Pao et al. [24]. Gianvecchio et al. [7] used a HOP system to detect bots in a famous MMOG (Massive Multiplayer Online Game) called World of Warcraft. They collected various types of information, including user trajectories, for use in bot detection. The trajectories of an FPS game and an MMOG are different because players have full control in the former, but not in the latter. Therefore, for the purposes of this study, it is more appropriate to apply our method to FPS games because they provide more user information.

## 3. Proposed Method

In this section, we describe the proposed method and explain how we deal with the verification and the recognition problems by exploiting the trajectory input. The method is implemented in three steps. The first step extracts useful features from the given trajectories. The second step measures the

[4]This work gives more systematic studies on more types of trajectories than the work in [24]; moreover, we clarify the difference between the verification and recognition tasks and give more studies on the recognition task on this work.

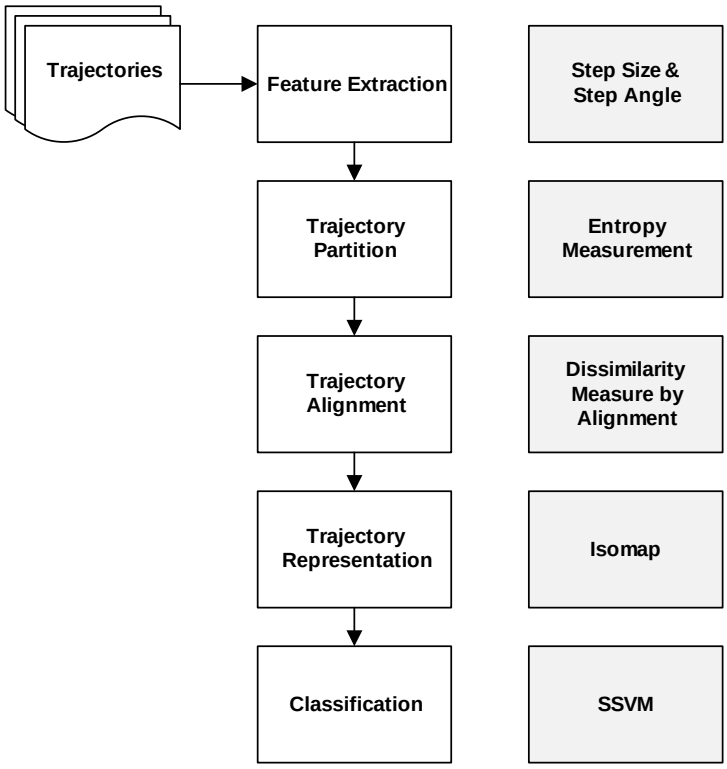



Figure 2: The framework of the proposed method: the left-hand side (white rectangles) shows the goals and the right-hand side (gray rectangles) shows the methods.

dissimilarity between a pair of trajectories. Then, based on the dissimilarity measure, the third step uses a manifold learning method called Isomap [29] to refine the dissimilarities and a classifier called Smooth SVM [13] to determine if the trajectories belong to authorized users or intruders. To improve the performance further, it is often helpful to substitute the dissimilarity measurement by two proceduces, *trajectory partition* followed by *trajectory alignment*. The framework of the method is shown in Figure 2. The left-hand side (white) shows the goals, and the right-hand side (gray) shows the meth-

ods used to achieve the goals. We discuss first the standard approach, then talk about the improvement by trajectory partition and trajectory alignment.

### *3.1. Feature Extraction*

Given a trajectory $\mathbf{s} = (\mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_T)$ of length $T$, we extract two types of features, namely, *steps* and *angles*. A step is a vector $\mathbf{x}_{t+1} - \mathbf{x}_t$. Based on that information, the Euclidean step size is computed by $\lambda_t = ||\mathbf{x}_{t+1} - \mathbf{x}_t||$; and, an angle $\theta_t$ is the angle between the vector $\mathbf{x}_{t+1} - \mathbf{x}_t$ and the $x$-axis.

### *3.2. Dissimilarity Measures*

We compute the dissimilarity measure for each pair of trajectories. Given a trajectory $\mathbf{s}$, let $\mathcal{M}_\mathbf{s}$ denote the model that best describes $\mathbf{s}$. In this work, we utilize a *continuous valued Markov chain* model[5] and assume Gaussianity in its transition to model a trajectory. There are two model parameters, $\sigma_\lambda$ and $\sigma_\theta$, which determine the transitions in the step size and angle respectively. To derive the parameter settings, we adopt the *maximum likelihood* principle. We indicate the model as $\mathcal{M}_\mathbf{s}(\sigma_\lambda, \sigma_\theta)$.

The step size parameter $\sigma_\lambda$ describes the standard deviation of step size $\lambda_{t+1}$, which we assume is centered in $\lambda_t$; and the angle parameter $\sigma_\theta$ describes the standard deviation of angle $\theta_{t+1}$, which we assume is centered in $\theta_t$. That is, based on the Markovian and Gaussianity properties of the coordinates of the consecutive time stamps $\mathbf{x}_t$, $\mathbf{x}_{t+1}$ and $\mathbf{x}_{t+2}$, we can express the probability

[5]The Markov chain (MC) model is popular in many learning tasks; for example, Shannon [28] uses it to model English text; Liò et al. [18] use it to model the DNA base substitution and amino acid replacement for phylogenetic reconstruction; and Gero et al. use it to model constructive memory [6].

density functions as follows:

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

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

Given a model $\mathcal{M}_{\mathbf{s}}$, the log-likelihood $\ell(\mathbf{s}; \mathcal{M}_{\mathbf{s}})$ of a trajectory $\mathbf{s}$ can be written as[6]

$$\begin{aligned}\ell(\mathbf{s}; \mathcal{M}_{\mathbf{s}}) &= \log L(\mathbf{s}; \mathcal{M}_{\mathbf{s}}) = \log\Big(p(\mathbf{x}_1)p(\mathbf{x}_2 \,|\, \mathbf{x}_1)\prod_{t=1} p(\mathbf{x}_{t+2} \,|\, \mathbf{x}_t, \mathbf{x}_{t+1})\Big), \\ &= \log P(\mathbf{x}_1) + \log p(\mathbf{x}_2 \,|\, \mathbf{x}_1) + \sum_{t=1} \log\Big(p(\mathbf{x}_{t+2} \,|\, \mathbf{x}_t, \mathbf{x}_{t+1})\Big), \quad (3)\end{aligned}$$

where $L$ is the likelihood function; and

$$\begin{aligned}\log p(\mathbf{x}_{t+2}|\mathbf{x}_t, \mathbf{x}_{t+1}) &= \log p(\lambda_{t+1} \,|\, \lambda_t) + \log p(\theta_{t+1} \,|\, \theta_t) \\ &= -\log(\sqrt{2\pi}\sigma_\lambda) - \frac{(\lambda_{t+1}-\lambda_t)^2}{2\sigma_\lambda^2} \\ &\quad -\log(\sqrt{2\pi}\sigma_\theta) - \frac{(\theta_{t+1}-\theta_t)^2}{2\sigma_\theta^2}. \quad (4)\end{aligned}$$

It is assumed that step size $\lambda_{t+1}$ and angle $\theta_{t+1}$ in time $t+1$ decide the distribution of the random variable $\mathbf{x}_{t+2}$ independently, given the previous two locations $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$ (i.e., given $\lambda_t$ and $\theta_t$). A variant of Eq. 4 can be written as

$$\log p(\mathbf{x}_{t+2}|\mathbf{x}_t, \mathbf{x}_{t+1}) = \alpha \log p(\lambda_{t+1} \,|\, \lambda_t) + (1-\alpha) \log p(\theta_{t+1} \,|\, \theta_t), \quad (5)$$

[6]The $p(\mathbf{x}_1)$ is a prior that assumes a uniform distribution, so it can be ignored in the computation of the maximum likelihood. We also assume that $p(\mathbf{x}_2 \,|\, \mathbf{x}_1)$ is a 2-D isotropic Gaussian $\frac{1}{\sqrt{2\pi}\sigma_\lambda} \exp\{-(\lambda_1 - \overline{\lambda})^2/(2\sigma_\lambda^2)\}$, centered in the origin, where $\overline{\lambda}$ is the mean of step size $\lambda_t$ in the trajectory.

where we would like to emphasize either the "influence" of step size or angle changes by choosing different values of $\alpha$ depending on cases.

Given the likelihood computation in Eq. 3, in our design, the dissimilarity (or distance) between two trajectories is measured by how well one of the trajectories is described by the model of the other trajectory. First, given a model $\mathcal{M}$, we compute the code length of a trajectory $\mathbf{s}$ (with respect to $\mathcal{M}$) as a negative logarithm of the likelihood, as follows:

$$c(\mathbf{s}|\mathcal{M}) = -\ell(\mathbf{s};\mathcal{M}) = -\log L(\mathbf{s};\mathcal{M})\,. \tag{6}$$

Note that $\mathcal{M}$ does not have to be the associated model of the trajectory $\mathbf{s}$. Based on the code length measure, we define the dissimilarity[7] between two trajectories $\mathbf{s}_1$ and $\mathbf{s}_2$ as follows:

$$d(\mathbf{s}_1,\mathbf{s}_2) = \frac{c(\mathbf{s}_1\,|\,\mathcal{M}_2) + c(\mathbf{s}_2\,|\,\mathcal{M}_1)}{c(\mathbf{s}_{12}\,|\,\mathcal{M}_{12})}\,, \tag{7}$$

where $\mathbf{s}_{12}$ is the trajectory that concatenates[8] $\mathbf{s}_1$ and $\mathbf{s}_2$, one after another, and $\mathcal{M}_{12}$ is the associated model of $\mathbf{s}_{12}$.

### *3.3. Preprocessing by Partition and Alignment*

Given a dataset of trajectories, we may find the dissimilarities of the trajectories by Eq. 7, and then apply the rest of the proposed steps (Isomap + SSVM) directly. However, to improve the performance further, it is often helpful to partition a single trajectory into sub-trajectories so that in each small sub-trajectory, we have "similar properties" or "stable parameters".

[7]The smaller the value, the closer will be the relationship between $\mathbf{s}_1$ and $\mathbf{s}_2$.

[8]The concatenation order, such as producing $\mathbf{s}_{12}$ by concatenating $\mathbf{s}_1$ followed by $\mathbf{s}_2$, or vice versa, makes very little difference. Only the concatenation point makes a difference.

We believe the partitioning makes the estimation of the step size changes and angle changes by Equations 1 and 2 more reliable. Next, we discuss the trajectory partition scheme and some post-processing steps that are necessary for the partitioned trajectories.

*3.3.1. Trajectory Partition*

If a trajectory contains different "properties" that cannot be modeled by a single set of parameters ($\sigma_\lambda$, $\sigma_\theta$) in Equations 1 and 2, we partition it into several sub-trajectories and estimate the model's parameters in each sub-trajectory. This technique provides a better description of the sub-trajectories and therefore the whole trajectory. We use the trajectory length to decide whether we need to partition the trajectory, and compute the entropy value to decide where to partition the trajectory if necessary.

Rigorously, given a trajectory $\mathbf{s}$, we make further partitions if $\mathbf{s}$ is longer than a pre-defined threshold $t$. If we need additional partitions, we use the entropy to decide the cutting point, which is the point with the lowest entropy in the partitioned sub-trajectory.

Given a trajectory $\mathbf{s}$, because the step size and angle independently decide the distribution of each step, the trajectory's entropy can be computed for step size and angle separately. That is, first, we discretize[9] the distribution $(\Delta\lambda, \Delta\theta)$ sampled from $(\Delta\lambda_t, \Delta\theta_t) = (\lambda_{t+1} - \lambda_t, \theta_{t+1} - \theta_t), t = 1 \ldots, T-1$,

[9]The distribution is in a discrete form when we collect the data. Here, discretization means that we combine several bins to form one group or we split bins into several smaller bins, depending on the discretization parameter. We use capital $P$ to denote that it is a probability mass function for the entropy computation.

and compute its entropy $H$ as follows (measured in bits):

$$
\begin{aligned}
H(\mathbf{s}) &= H(\Delta\lambda, \Delta\theta) \\
&= -\sum_{i,j} P(\Delta\lambda_i, \Delta\theta_j) \log P(\Delta\lambda_i, \Delta\theta_j) \\
&= -\sum_{i,j} P(\Delta\lambda_i) P(\Delta\theta_j) \log P(\Delta\lambda_i) P(\Delta\theta_j) \qquad (8) \\
&= -\sum_{i} P(\Delta\lambda_i) \sum_{j} P(\Delta\theta_j) \left(\log P(\Delta\lambda_i) + \log P(\Delta\theta_j)\right) \\
&= -\sum_{i} P(\Delta\lambda_i) \sum_{j} P(\Delta\theta_j) \log P(\Delta\lambda_i) \\
&\quad -\sum_{i} P(\Delta\lambda_i) \sum_{j} P(\Delta\theta_j) \log P(\Delta\theta_j) \\
&= -\sum_{i} P(\Delta\lambda_i) \log P(\Delta\lambda_i) - \sum_{j} P(\Delta\theta_j) \log P(\Delta\theta_j) \\
&= H(\Delta\lambda) + H(\Delta\theta)\,, \qquad (9)
\end{aligned}
$$

where $i$ and $j$ are indices for the discretized distributions $\Delta\lambda$ and $\Delta\theta$ respectively; and Eq. 8 is derived from the independence of $\Delta\lambda$ and $\Delta\theta$. Given a cut point, we use Eq. 9 to compute the entropy values of the two sub-trajectories, and compute their weighted summation (weighted by the sub-trajectory length). We then choose the best cut point to be the point where we have the lowest weighted entropy. Similar to Eq. 5, we can also give a variant of Eq. 9 as

$$
H(\mathbf{s}) = \alpha H(\Delta\lambda) + (1-\alpha) H(\Delta\theta)\,. \qquad (10)
$$

Figure 3 shows the results of partitioning a trajectory. The blue line is the trajectory input, and red circles indicate the cutting points. Figures 3 (a1)-(a3) are based on synthesized trajectories that combine half circles and

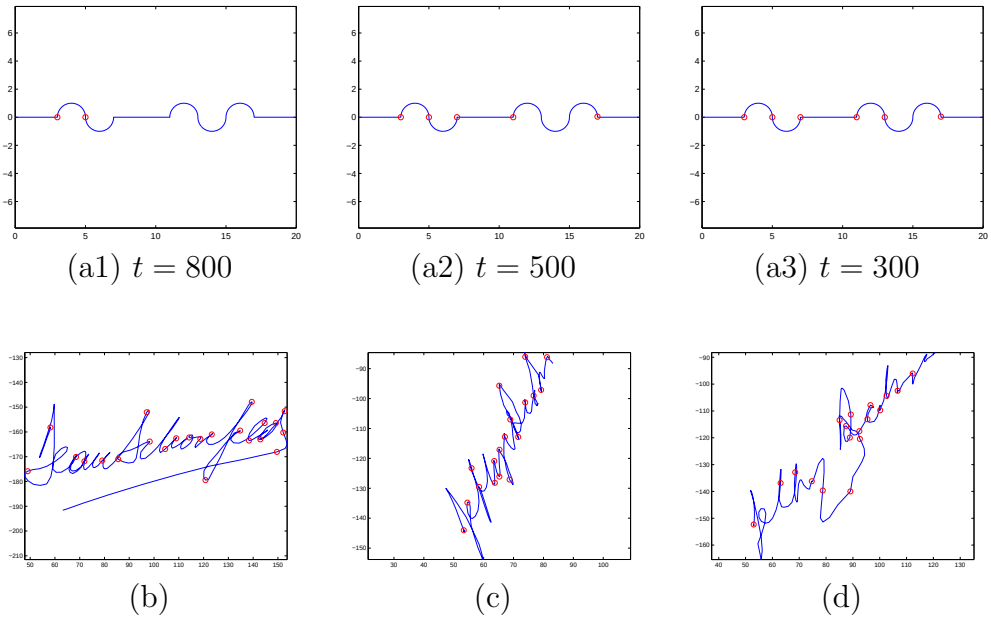



Figure 3: (a1)-(a3): the different thresholds $t$ used to partition a trajectory; (b), (c), and (d): the trajectory partitions for different handwritten signatures.

straight lines. We tested different threshold values to partition the trajectory. (a3) appears to have more partitions than (a1), because it allows no trajectory longer than 300. We observe that the cutting points gradually catch the curve property from (a1) to (a3). Figures 3 (b), (c), and (d) show the results when we apply the proposed trajectory partition algorithm to real handwritten data (see Section 4.1.2 for more detail). The red splitting points almost partition on inflection points, or points that have high curvature.

### *3.3.2. Trajectory Alignment*

The output of trajectory partitioning is a sequence of sub-trajectories. Given two trajectories $\mathbf{s}_1$ and $\mathbf{s}_2$, and their respective partitioned sets $\mathbf{s}_{11}, \mathbf{s}_{12}, \ldots, \mathbf{s}_{1u}$, $\ldots, \mathbf{s}_{1U}$ and $\mathbf{s}_{21}, \mathbf{s}_{22}, \ldots, \mathbf{s}_{2v}, \ldots, \mathbf{s}_{2V}$, respectively, we perform trajectory align-

ment to find the dissimilarity $D$ between $\mathbf{s}_1$ and $\mathbf{s}_2$. The alignment is executed by a dynamic programming procedure in which the recurrence value $D(u, v)$ describes the dissimilarity between $\mathbf{s}_{11}, \ldots, \mathbf{s}_{1u}$ and $\mathbf{s}_{21}, \ldots, \mathbf{s}_{2v}$ as follows:

$$D(u, v) = 0 \quad \text{if } u = 0 \text{ or } v = 0\,,$$

and if $u > 0, v > 0$,

$$D(u, v) = \min \begin{cases} D(u-1, v-1) + \frac{1}{2}(|\mathbf{s}_{1u}| + |\mathbf{s}_{2v}|)\delta(\mathbf{s}_{1u}, \mathbf{s}_{2v}) & \text{match} \\ D(u-1, v) + |\mathbf{s}_{1u}|\delta(\mathbf{s}_{1u}, -) & \text{gap} \\ D(u, v-1) + |\mathbf{s}_{2v}|\delta(-, \mathbf{s}_{2v}) & \text{gap} \end{cases}, \quad (11)$$

where $|\mathbf{s}|$ denotes the length of trajectory $\mathbf{s}$. The dissimilarity function $\delta$ for two sub-trajectories $\mathbf{s}_{1u}$ and $\mathbf{s}_{2v}$, or for a sub-trajectory and a "gap" (denoted by "$-$") is given by

$$\delta(\mathbf{s}_{1u}, \mathbf{s}_{2v}) = \frac{c(\mathbf{s}_{1u}|\mathcal{M}_{2v}) + c(\mathbf{s}_{2v}|\mathcal{M}_{1u})}{c(\mathbf{s}_{1u2v}|\mathcal{M}_{1u2v})\,}, \quad (12)$$

$$\delta(\mathbf{s}_{1i}, -) = \delta(-, \mathbf{s}_{2j}) = G\,, \quad (13)$$

where $\mathbf{s}_{1u2v}$ is the concatenated trajectory of trajectory $\mathbf{s}_{1u}$ and trajectory $\mathbf{s}_{2v}$; and $\mathcal{M}_{1u2v}$ is the Markov model associated with $\mathbf{s}_{1u2v}$. We define the dissimilarity between $\mathbf{s}_1$ and $\mathbf{s}_2$ as

$$D(\mathbf{s}_1, \mathbf{s}_2) = D(U, V)\,. \quad (14)$$

Note that Eq. 12 is very similar to Eq. 7, where we compute the dissimilarity between two original (non-partitioned) trajectories. In fact, Eq. 14 and Eq. 7 give identical result if no partition is performed and we use the input of the same trajectory length[10]. We apply a linear gap penalty, as shown in

[10]In this work, the input trajectories that we compute their dissimilarities are always in the same length.

Eq. 13, which allow gaps to exist in the match of two trajectories that have *different* numbers of sub-trajectories. In this work, the gap penalty is set to be 30 at all times for simplicity. Note that dynamic programming alignment usually incurs *quadratic* complexity for the computation; however, because the number of sub-trajectories is small in this case, the whole computation can be completed very quickly.

### *3.4. Trajectory Representation and Labeling*

Given the pairwise dissimilarities of trajectories derived by Eq. 14 (or Eq. 7 if no partition is necessary), we could utilize a simple method, such as the $k$ nearest neighbor ($k$NN) algorithm, to determine if a trajectory is similar to the trajectories of the real account owner or if it belongs to an intruder. However, we would like to design a more effective method, by an improvement based on the concept of *manifold learning*. We seek an embedding feature space to represent a set of trajectories; and in the feature space, we use the Smooth SVM classifier [13] to label the trajectories [16]. In the feature space, two trajectories are regarded as close (similar) if (1) they have a small dissimilarity score, computed by Eq. 14; or, (2) both of them are close (similar) to a third trajectory. We utilize Isomap [29] to find the feature space to represent the trajectories. The steps are as follows.

1. Construct a neighborhood graph by linking each pair of trajectories/points that qualify as neighbors.
2. Find the shortest path between each pair of points and take it as the approximation of their geodesic distance.
3. Take the pairwise (geodesic) distances as the input and apply Multidimensional Scaling (MDS) [4] to find the global Euclidean coordinates of

the points.

Figure 5 shows some examples of embedding in a 2-D space, after applying Isomap. Note that the "optimal" dimensionality (also called the *intrinsic dimensionality*), where we can separate different kinds of trajectories effectively, is not necessarily two dimensions. The intrinsic dimensionality can be estimated by finding the "elbow" point in the residual variance curve [29]. Ideally, we should be able to use any classifier in the feature space to determine whether a trajectory belongs to the true account owner or an unauthorized person. In this study, we use SSVM [13] to evaluate the performance of proposed method.

## 4. Experiments

The experiments were divided into two parts. They focused on the verification and recognition tasks respectively. The verification task is the main focus in this work. The performance of our method in terms of the recognition task was only assessed in the situation where the number of trajectory identities was small. In real cases, the number of identities may range from hundreds to millions. Analyzing such cases is beyond the scope of the present study. We evaluated the performance of the proposed method on the verification and recognition tasks given inputs of different kinds, including on-line game traces, handwriting traces, mouse traces, and animal movement traces. We showed that the proposed method is superior to or competitive to existing state-of-the-art approaches in related applications. Moreover, we studied the effectiveness of the proposed method given inputs of different length. Before discussing the experiment results, we introduce the datasets used in the

Table 1: Statistics of the datasets used in the verification and recognition experiments, and the associated parameters. In the table, $k_{Iso}$ denotes the number of neighbors $k$ used by $k$NN to construct the neighborhood graph for Isomap. For simplicity and to ensure a fair comparison, it is assumed that the intrinsic dimensionality of all the datasets is 5. We use the length threshold to decide when we need to partition trajectories. The $\alpha$ in Eq. 5 and Eq. 10 is set to 0.5 at all times except that for the animal dataset we set $\alpha = 0.7$.

| | Name | Classes | Instances | Trace Length | $k_{Iso}$ | Intrinsic Dim. | Length Threshold |
|---|---|---|---|---|---|---|---|
| Veri. | Handwriting | 11 | 110 | 702 | 7 | 5 | 200 |
| | Mouse | 14 | 217 | 16665 | 6 | 5 | 300 |
| | $\text{Game}_v$ | 94 | 940 | 1000 | 8 | 5 | 300 |
| Recog. | $\text{Game}_r$ | 4 | 173 | 1000 | 5 | 5 | 300 |
| | Animal | 3 | 101 | 145 | 5 | 5 | 160 |

experiments.

### *4.1. Data Description*

Our datasets include real traces of on-line game players, handwriting traces, mouse traces, and animal movement traces in their natural environments. Table 1 shows the statistics of the datasets as well as the parameters settings used in the experiments.

*4.1.1. Game Trajectory*

The game trace dataset is comprised of avatar movements obtained from Quake 2, a popular FPS game developed by id Software[11]. In the game, each player controls an avatar's movements directly. The objective is to kill enemies and accumulate treasure to achieve a high score. The dataset comprises human traces and traces from well-known game bots (automatic programs) such as CR Bot[12], Eraser Bot[13] and ICE Bot[14]. To obtain a comprehensive and fair benchmark, we downloaded human traces from some public web sites. More details can be found in [2, 3]. In total, the human and bot traces last 143.8 hours. For the verification experiment, we sort the human traces into 94 users, called the $\text{Game}_v$ dataset; and for the recognition experiment, we sort all traces into four types of users (one human group and three groups of bots), called the $\text{Game}_r$ dataset.

*4.1.2. Handwritten Signature*

The handwriting dataset[15] from SVC 2004 handwritten signature verification competition is a benchmark for user verification. It includes signature samples written for 80 subjects, 40 from the first set (Set 1) and the other 40 from the second set (Set 2). In each subject, we have 20 genuine signatures and 20 skilled imitations written by other forgers. In total, there are 3200 signatures in the dataset. In the dataset, a data record (i.e. a trace) is a

[11]http://www.idsoftware.com/

[12]http://arton.cunst.net/quake/crbot/

[13]R. R. Feltrin. Eraser bot 1.01

[14]http://ice.planetquake.gamespy.com/

[15]http://www.cse.ust.hk/svc2004/download.html

sequence of 2-D coordinates in a period of $T$ seconds, $(x_t, y_t)$, $t = 1, \ldots, T$. In the experiment, given a subject, the goal is to correctly label the signatures to either genuine signatures or the signatures written by forgers.

#### *4.1.3. Mouse Movement Trajectory*

We also compiled a mouse trace dataset containing the mouse traces of 14 users when they work on their personal computers. For each trace that lasts for more than 30 minutes, we randomly choose a 30-minute piece as our data. In total, the dataset contains 217 instances. Given the mouse movement, we may observe various patterns when users are involved in different activities at different times. For instance, while one user plays an FPS game, another may surf web sites and listen to music simultaneously. If we compare these two cases, we may find that the first user moves the mouse continuously, while the second user moves it infrequently.

#### *4.1.4. Animal Trajectory*

The above datasets were collected via computers. We also investigate the proposed method's performance on the dataset of animal movement traces complied by Lee et al. [12]. The traces were provided by the Starkey project[16], which started in 1989. The set contains the movement trajectories of three kinds of animals: elk, mule deer, and cattle. In our experiment, we focus on the data collected in June 1995. Specifically, there are 38, 30, and 34 instances of elk, mule deer, and cattle movements respectively. For our pur-

[16]http://www.fs.fed.us/pnw/starkey/index.shtml. Wisdom, Michael J. 1988. "The Starkey Project: deer and elk research for the future". Oregon Chapter, The Wildlife Society, Pendleton, OR., U.S.A.

Table 2: Summary of the verification results for various inputs. The table shows the average error rates (in percentages) of the SSVM classification after performing ten-fold cross-validation three times.

| Data Set | Training Error | Test Error |
|---|---|---|
| Handwriting | 1.18 | 3.91 |
| Mouse | 6.67 | 8.10 |
| Game | 10.80 | 15.62 |

poses, only the $x$ and $y$ coordinates in the data are used.

### *4.2. Verification*

We design the verification experiment as follows. First, given all trajectories, based on the dissimilarity measure in Eq. 14 (or a simpler version Eq. 7), we utilize Isomap to find the representation space for all the trajectories. Second, given two identities (a true account owner and an intruder), we select all trajectories belonging to the two identities in the representation space, then we operate a binary classification (under ten-fold cross-validation, for three repeats) to label the trajectories as belonging to the true account owner or not.

We calculate the *training error* and the *test error* in the following ways respectively for our evaluation metrics. The training error is the error rate on the training set (nine out of ten parts in the ten-fold cross-validation process), and the *test error* is the error rate on the test set (the rest). Given four types of classified data: true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN), the error rate is define by $ER =$

Table 3: Comparison of the proposed method to previous methods for handwritten signature verification. The table shows the error rates (in percentages). The verification by the proposed method is done by SSVM binary classification, after performing ten-fold cross-validation three times. The numbers of other methods, including normalized Position, directional Angles, shape context (SC), on-line context (OC), OLT are copied from Qiao et al. (QLT method) [25]. All methods except QLT are on-line methods, and QLT tried to find the on-line information from off-line inputs. The proposed method is very close to the best one among all on-line methods. The comparison to off-line methods (other than QLT) is not shown because most off-line methods perform worse than on-line methods. More information about the compared methods can be found in [25].

| Data Set | Position | Angle | SC | OC | QLT | Ours Training | Ours Test |
|---|---|---|---|---|---|---|---|
| Set 1 | 13.6 | 6.5 | 7.2 | 5.8 | 7.3 | 3.29 | 4.21 |
| Set 2 | 11.9 | 6.3 | 4.9 | 4.6 | 7.4 | 2.42 | 4.93 |

$(FP + FN)/(FP + FN + TP + TN)$. If there are $n$ identities, we need to run $n(n-1)/2$ different identity combinations to get an average result for a fair evaluation.

Table 2 shows the performance of proposed method on different kinds of trajectories. As expected, verification of the handwritten trajectory dataset achieves the best error rate (3.91%), followed by verification of the mouse trace (8.10%), and verification of the game trace (15.62%). Handwriting traces give us the best discriminative power because they are based on finer motions. In contrast, game traces are usually collected in a restricted environment, so they lack some degree of freedom in their movement to show the true identity of the trace owners.

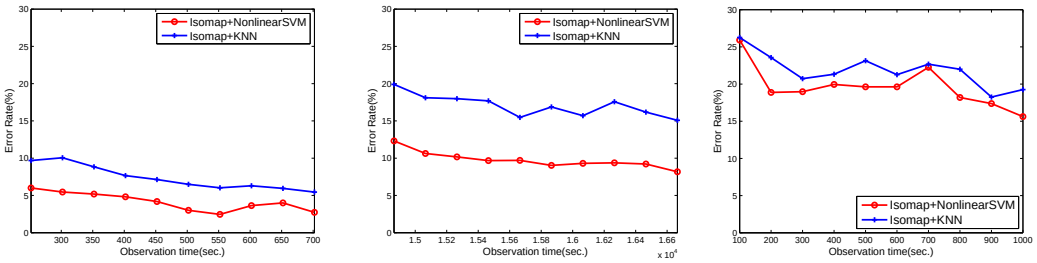



Figure 4: The test error rates obtained by applying two detection schemes, SSVM and $k$NN, to: (a) handwriting traces, (b) mouse traces and (c) on-line game traces, of different length. All verification errors decrease over time; also, the SSVM-based method outperforms the $k$NN-based method.

To compare to existing methods, we focus on the handwritten signature verification because it is one the most emphasized verification tasks in the past. In Table 3, we compare the proposed method to various existing methods that used on-line information for user verification. The result show that the proposed method nearly outperforms all other methods (just slightly worse than SC and OC in Set 2). Thanks to the novel dissimilarity measure and Isomap tuning, the proposed method can effectively separate input trajectories of different patterns for verification.

*Using Trajectories of Different Length*

Since we want to verify the user's identity as early as possible given a trajectory, we are interested in analyzing the performance when only a shorter input trajectory is given, rather than wait for a longer one. In Figure 4, given inputs such as handwriting traces, mouse traces, or on-line game traces, of different length, we show the error rates of verification task for SSVM

and $k$NN. We use both SSVM and $k$NN as the classifiers after finding the representation by Isomap. The verification results (i.e., whether the trace belongs to the true account owner or not) can be summarized as follows. First, the verification errors on all types of inputs decrease when we use longer traces. Second, the SVM-based method outperforms the $k$NN-based method. Third, similar to the result in Table 2, handwriting trajectory dataset again achieves the best verification performance, followed by verification of the mouse traces, and verification of the game traces. In particular, we observe that for handwriting traces, after several hundred seconds, such as after observing 550 sampling points, the (nonlinear) SVM classifier yields as low as 2-4% error rate.

### *4.3. Recognition*

In the recognition task, we try to find the true identity of the user from the given trajectories. We focus on the case where the number of identities is small. Working on a large-scale database is challenging, especially when the inputs, such as mouse traces and on-line game traces, only have weak discriminative power. Even so, we would like to study the possibility of applying the proposed method to the recognition problem. We consider two datasets. The first one is used to distinguish between human players and various kinds of bots; and the second is used to distinguish between the movements of different animals. Using different datasets between the verification and recognition tasks is simply because: (1) the datasets used for verification task involves large numbers of identities that are beyond the discriminative power of the proposed method at this moment; (2) we need particular benchmark datasets that we do have past studied results to compare the performance.

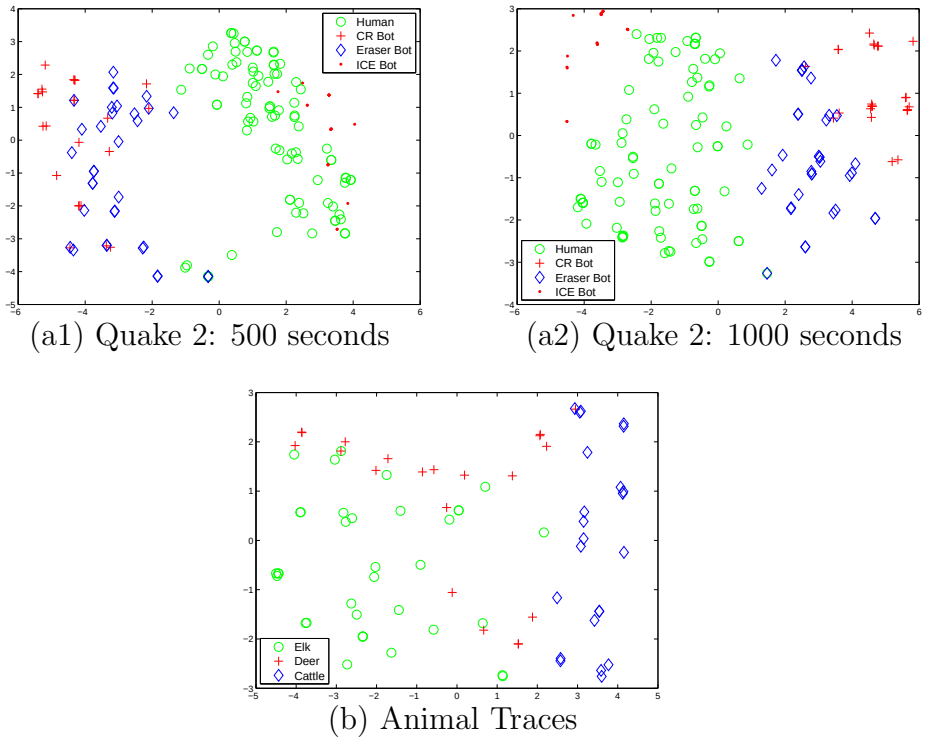
Table 4: The recognition results on the $\text{Game}_r$ dataset with inputs of different length. The table shows the average error rates (in percentages) of the SSVM classification when ten-fold cross-validation is performed three times. The dataset includes four classes: human players and three types of bots (CR Bot, Eraser Bot, and ICE Bot).

| Trace Length | Training Error | Test Error |
|---|---|---|
| 500 seconds | 5.29 | 7.07 |
| 1000 seconds | 1.89 | 2.53 |

We again use training error and test error based on a ten-fold cross-valiation process to be the evaluation metrics. The error rate here simply indicates the percentage of data that are wrongly classified by the proposed method, such as classifying human as CR Bot given the game trajectories, or classifying elk as cattle given the animal trajectories.

*4.3.1. Game Trajectory*

To further demonstrate the effectiveness of the proposed method, we use it to extract different patterns from human traces and three types of bot traces, i.e., it is a multi-class classification problem. The results are shown in Table 4. On a 500-second trace, our method can achieve 7.07% error rate; and on a 1000-second trace, the error rate decreases to 2.53%. From Figures 5 (a1) and (a2), we observe that there is better separation between classes when longer trajectories are collected. Obviously, we prefer a method that can identify user behavior as quickly as possible, before account information can be stolen by hackers.




(a1) Quake 2: 500 seconds (a2) Quake 2: 1000 seconds

(b) Animal Traces

Figure 5: (a1) (a2) given inputs of different length, the representations of the Quake 2 traces after projection by Isomap into a 2-D space, where a point represents a human trace (green circles) or a bot trace (other symbols); and (b) the representation of the animal traces, after projection by Isomap into a 2-D space. The $x$- and $y$-axes are the first and second principal coordinates from Isomap. Classification is usually performed in a higher dimensional space, called the space of *intrinsic dimensionality*, as shown in Table 1.

### *4.3.2. Animal Trajectory*

Next, we consider a dataset of animal movement trajectories [12]. There are three kinds of animal trajectories for classification[17]. Figure 5 (b) shows

[17]In the SSVM classification, we adopt a hierarchical approach to solve the multi-class classification problem. The elk and deer are combined to form one group for the first binary classification, followed by another binary classification to separate them.

Table 5: Error rates of the proposed method and Lee et al.'s *TraClass* [12] for the recognition task on animal movement traces. For the proposed method, the table shows the average error rates (in percentages) by SSVM classification, when ten-fold cross-validation is performed three times. We set the threshold as $t = 160$. Clearly, our method outperforms the *TraClass* method.

| | Our Method | | *TraClass* |
|---|---|---|---|
| Data Set | Training Error | Test Error | Test Error |
| Animal | 7.53 | 12.51 | 16.70 |

the embeddings after applying Isomap. Interestingly, the cattle traces (blue diamonds) are clearly separated from the elk traces (green cycles) and deer traces (red crosses). This confirms the biological relationship between those different traces. Moreover, we show that the proposed method outperforms the *TraClass* [12]. As shown in Table 5, the error rate of recognition task under our method is 12.51%, whereas *TraClass* only achieves 16.70% in its error rate.

To summarize, the study of the game and animal movement datasets suggests that, based on the analysis of the patterns hidden in the trajectories, the proposed method is capable of separating different kinds of trajectories. The general recognition problem, which involves recognizing the user without being given a pre-defined identity, is still difficult to solve[18]. However, we

[18]It is difficult, especially when the number of individuals in the database is large, e.g., thousands or millions of individuals. Although identifying the true account owner based purely on the trajectory input without any other information is difficult, the proposed method provides a possible solution.

believe that the proposed method could provide the basis for further studies.

### *4.4. Computation Time*

The proposed method includes several steps as in Figure 2. First, we compute the pairwise dissimilarity between each pair of trajectories. Sometimes we also need to compute the trajectory entropy for partition and align the sub-trajectories. Following that, we compute the Isomap representations. Given the dataset size from a few hundreds to fewer than one thousand instances, the whole procedure above takes less than a few seconds to finish on average. After obtaining the Isomap representations, we apply SSVM for classification, under a ten-fold cross-validation procedure. In each fold and each pair of identities for classification, the training takes around or less than 2 seconds while the test procedure takes less than one second on average. The computation time simply shows that we can easily apply the proposed method to real verification and recognition tasks for similar size of datasets. The whole computation is done on a regular Pentium-5 machine with a 32-bit Microsoft Windows operating system.

## 5. Conclusion

In this work, we have proposed a novel method for user trajectory verification and recognition. The scheme is based on a dissimilarity measure combined with a manifold learning adjustment by Isomap. To compute the dissimilarity measure, we employ a Markov chain model to describe the behavior of the target trajectory. We have applied our method to various types of trajectories, including handwriting traces, mouse traces, game traces, and traces of animal movements in their natural environments. The results show

that the trajectory input contains sufficient information for verification; and our method is effective in identifying the hidden patterns embedded in trajectories. The proposed method is also capable of solving related recognition problems, such as recognizing an account owner without a pre-defined account identity as long as the number of identities is not large. Moreover, the proposed method outperforms or is comparable to the existing state-of-the-art approaches for both of the verification and recognition tasks, without any extra action from users. Thus, we believe that the proposed method merits further investigation by researchers interested in account security and related fields.

## References


[1] Abdul Rahim Ahmad, M. Khalia, C. Viard-Gaudin, and E. Poisson. Online handwriting recognition using support vector machine. In *Second International Conference on Artificial Intelligence in Engineering and Technology*, volume A, pages 311–314, nov. 2004.

[2] Kuan-Ta Chen, Andrew Liao, Hsing-Kuo Pao, and Hao-Hua Chu. Game bot detection based on avatar trajectory. In *Proceedings of IFIP ICEC*, pages 94–105, 2008.

[3] Kuan-Ta Chen, Hsing-Kuo Pao, and Hong-Chung Chang. Game bot identification based on manifold learning. In *Proceedings of ACM NetGames*, pages 21–26, October 2008.

[4] Trevor F. Cox and Michael A. A. Cox. *Multidimensional Scaling, Second Edition.* Chapman & Hall/CRC, 2000.

[5] Hamido Fujita, Jun Hakura, and Masaki Kurematu. Intelligent human interface based on mental cloning-based software. *Know.-Based Syst.*, 22:216–234, April 2009.

[6] John S. Gero and Wei Peng. Understanding behaviors of a constructive memory agent: A markov chain analysis. *Know.-Based Syst.*, 22:610–621, December 2009.

[7] Steven Gianvecchio, Zhenyu Wu, Mengjun Xie, and Haining Wang. Battle of botcraft: fighting bots in online games with human observational proofs. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *ACM Conference on Computer and Communications Security*, pages 256–268. ACM, 2009.

[8] Chien-Ju Ho, Chen-Chi Wu, Kuan-Ta Chen, and Chin-Luang Lei. Deviltyper: A game for CAPTCHA usability evaluation. *ACM Computers in Entertainment*, 9(1):3:1–3:14, April 2011.

[9] Anil K. Jain, Patrick Flynn, and Arun A. Ross. *Handbook of Biometrics*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[10] Anil K. Jain, Friederike D. Griess, and Scott D. Connell. On-line signature verification. *Pattern Recognition*, 35(12):2963–2972, 2002.

[11] Eamonn Keogh, Stefano Lonardi, and Chotirat Ann Ratanamahatana. Towards parameter-free data mining. In *KDD '04: Proceedings of the tenth ACM SIGKDD inter. conf. on Knowledge discovery and data mining*, pages 206–215, New York, NY, USA, 2004. ACM.

[12] Jae-Gil Lee, Jiawei Han, Xiaolei Li, and Hector Gonzalez. *TraClass*: trajectory classification using hierarchical region-based and trajectory-based clustering. *PVLDB*, 1(1):1081–1094, 2008.

[13] Yuh-Jye Lee and O. L. Mangasarian. SSVM: A smooth support vector machine for classification. *Comput. Optim. Appl.*, 20(1):5–22, 2001.

[14] M. Li, J. H. Badger, X. Chen, S. Kwong, P. Kearney, and H. Zhang. An information-based sequence distance and its application to whole mitochondrial genome phylogeny. *Bioinformatics*, 17(2):149–154, 2001.

[15] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications (2nd Ed.)*. Springer, New York, 1997.

[16] Fengyi Lin, Ching-Chiang Yeh, and Meng-Yuan Lee. The use of hybrid manifold learning and support vector machines in the prediction of business failure. *Knowledge-Based Systems*, 24(1):95 – 101, 2011.

[17] Jessica Lin, Eamonn J. Keogh, Stefano Lonardi, and Bill Yuan-Chi Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *DMKD*, pages 2–11, 2003.

[18] Pietro Liò and Nick Goldman. Models of molecular evolution and phylogeny. *Genome Res*, 8:1233–1244, 1998.

[19] Ronald Metoyer, Simone Stumpf, Christoph Neumann, Jonathan Dodge, Jill Cao, and Aaron Schnabel. Explaining how to play real-time strategy games. *Know.-Based Syst.*, 23:295–301, May 2010.

[20] Greg Mori and Jitendra Malik. Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA. In *Computer Vision and Pattern Recognition*, volume 1, pages 134–141, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[21] Mario E. Munich and Pietro Perona. Visual identification by signature tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(2):200–217, 2003.

[22] Hsing-Kuo Pao and John Case. Computing entropy for ortholog detection. In *International Conference on Computational Intelligence*, pages 89–92, 2004.

[23] Hsing-Kuo Pao, Kuan-Ta Chen, and Hong-Cheng Chang. Game bot detection via avatar trajectories analysis. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(3):162–175, September 2010.

[24] Hsing-Kuo Pao, Hong-Yi Lin, Kuan-Ta Chen, and Junaidillah Fadlil. Trajectory based behavior analysis for user verification. In *IDEAL*, pages 315–322, 2010.

[25] Yu Qiao, Jianzhuang Liu, and Xiaoou Tang. Offline signature verification using online handwriting registration. In *CVPR*, 2007.

[26] Jonas Richiardi and Andrzej Drygajlo. Gaussian mixture models for on-line signature verification. In *WBMA '03: Proceedings of the 2003 ACM SIGMM workshop on Biometrics methods and applications*, pages 115–122, New York, NY, USA, 2003. ACM.

[27] Bruce Schneier. Two-factor authentication: too little, too late. *Commun. ACM*, 48:136, April 2005.

[28] C. E. Shannon. A mathematical theory of communication. *Bell Syst Tech. J.*, 27:379–423, 1948.

[29] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000.

[30] Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. CAPTCHA: Using hard AI problems for security. In *EUROCRYPT*, pages 294–311, 2003.