Game Bot Detection via Avatar Trajectory Analysis

Hsing-Kuo Pao, Kuan-Ta Chen, *Member, IEEE*, and Hong-Chung Chang

***Abstract*—The objective of this work is to automatically detect the use of game bots in online games based on the trajectories of account users. Online gaming has become one of the most popular Internet activities in recent years, but cheating activity, such as the use of game bots, has increased as a consequence. Generally, the gaming community disapproves of the use of bots, as users may obtain unreasonable rewards without making corresponding efforts. However, game bots are hard to detect because they are designed to simulate human game playing behavior and they follow game rules exactly. Existing methods cannot solve the problem as the differences between bot and human trajectories are generally hard to describe. In this paper, we propose a method for detecting game bots based on some dissimilarity measurements between the trajectories of either bots or human users. The measurements are combined with manifold learning and classification techniques for detection; and the approach is generalizable to any game in which avatars' movements are controlled by the players directly. Through real-life data traces, we observe that the trajectories of bots and humans are very different. Since certain human behavior patterns are difficult to mimic, the characteristic can be used as a signature for bot detection. To evaluate the proposed scheme's performance, we conduct a case study of a popular online game called *Quake 2*. The results show that the scheme can achieve a high detection rate or classification accuracy on a short trace of several hundred seconds.**

***Index Terms*—Behavior analysis, bot detection, cheating, manifold learning, online games, similarity measure, trajectory.**

I. Introduction

THE objective of this work is to automatically detect the use of game bots in online games based on the trajectories of account users. Although humans can easily detect game playing bots, as exhibited in the competition The 2K BotPrize,[1] it is shown to be difficult to design an automatic mechanism for detecting such bots [1], [2]. By analyzing the behavior patterns hidden in the trajectories, we want to determine whether an unseen input is a bot or a human user. Online gaming is now one of the most popular Internet activities; however, as the population of online gamers has increased, game cheating problems, such as the use of *game bots*, have become more serious. Game bots are automated programs, with or without artificial intelligence, which help players enhance, accelerate, or bypass some routines in a game. For example, in first-person shooter (FPS) games, users can employ bots to play in place of themselves in order to get high scores and gain a reputation in the community. Similarly, in massively multiplayer online role player games (MMORPGs), players can save a great deal of time by using bots to perform repetitive tasks, such as slashing low-level monsters, or fishing in a river to master the avatar's fishing skills [1]. The use of bots in MMORPGs is notoriously related to "gold farming" [3], wherein bots are used to harvest the resources in the game world, accumulate the experience points and fortune for game characters, and later sell the resources and even the characters (and the associated accounts) to genuine players in exchange for in-game currency or real-world money. This phenomenon often annoys honest users as it erodes the balance and order of the game world, where bot users and the customers of gold farmers can monopolize scarce resources and easily outperform honest users in terms of economics and military force. This problem is second only to account theft [4] in their impact upon such games.

Generally, the gaming community disapproves of the use of game bots, as bot users obtain unreasonable rewards without corresponding efforts. However, game bots are hard to detect because they are designed to simulate human game playing behavior and they follow game rules exactly. Some bot detection studies [2], [5], [6] propose using CAPTCHA tests during a game to determine whether an avatar is actually controlled by a person. Although this method is effective, it disrupts the game play and degrades players' feelings of immersion in the virtual world [7], [8]. Alternatively, passive detection approaches, such as schemes based on traffic analysis [1] and schemes based on avatars' shooting accuracy in FPS games [9], have been proposed. The drawbacks of these schemes are that the former assumes a game bot works as a standalone client, while the latter are only suitable for detecting aim bots in shooting games.

In this paper, we propose a general approach for all genres of games in which players control an avatar's movements directly. Taking the avatar's movement trajectory as the input, we adopt a learning method for bot detection. By analyzing a trajectory, we determine whether a behavior pattern belongs to a particular player and can therefore be taken as the *signature* of the player. The rationale behind our approach is that the trajectory of an avatar controlled by a human player is hard to simulate. Players control the movement of avatars based on their knowledge, experience, intuition, and a great deal of environmental

Manuscript received November 20, 2009; revised May 26, 2010; accepted August 16, 2010. Date of publication September 02, 2010; date of current version September 15, 2010. This work was supported in part by Taiwan Information Security Center (TWISC), National Science Council under Grants NSC98-2221-E-011-105 and NSC98-2219-E-011-001 and in part by Taiwan E-learning and Digital Archives Programs (TELDAP) sponsored by the National Science Council of Taiwan under Grants NSC98-2631-001-011 and NSC98-2631-001-013.

H.-K. Pao is with the Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei 106, Taiwan (e-mail: pao@mail.ntust.edu.tw).

K.-T. Chen is with the Institute of Information Science, Academia Sinica, Taipei 115, Taiwan (e-mail: ktchen@iis.sinica.edu.tw).

H.-C. Chang was with the Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei 106, Taiwan. He is now with the Institute for Information Industry (III), Taipei 10622, Taiwan (e-mail: chz1971@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCIAIG.2010.2072506

[1]http://botprize.org/

information provided in the game. Since human decisions are sophisticated and depend on multitudinous observable and unobservable factors, how to model and simulate realistic movements is still an open question in the AI field.

To detect game bots, it is necessary to summarize the information about the behavior pattern hidden in the avatar's trajectory. Then, the pattern can be used as a signature for bot detection based on classification techniques. Treating the trajectory as a long series of 2-D or 3-D coordinates (depending on whether the game is 2-D or 3-D) may be unwise, as the data lies in a very high-dimensional space and we may face the *curse of dimensionality* problem [10]. Alternatively, we can transform the trajectory into a pattern in a low-dimensional feature space and detect bots in that space. We propose two *dissimilarity measures* to describe the relationships between trajectories, and combine the measures with a *manifold learning* approach called Isomap [11] for dimension reduction. Then, we use the trajectories represented in the low-dimensional space as input for classification and bot detection.

A naive approach takes the Euclidean distance as the dissimilarity measure between two sets of trace data. However, the proposed dissimilarity measures achieve more robust detection. The first measure is based on the Kullback–Leibler (KL) divergence [12] between two step-size distributions derived from trajectories. The second considers the temporal information by building associated Markov chain models of the sequences. It then assesses how well one sequence is described by the model associated with the other sequence to determine their distance/dissimilarity. In this work, each trace is described by a continuous-valued Markov chain with Gaussian-distributed transitions on step-size changes and angle changes. If two traces are similar, we expect to find a short description code of one trace given the model for the other trace, and *vice versa*. We adopt *Quake 2* as our case study because it is a classic and popular FPS game, and many real-life human traces are available on the Internet. Therefore, we can use such traces to validate our proposed scheme.

The contribution of this paper is threefold. 1) We propose using a manifold learning framework to detect game bots based on avatars' trajectories. The model is generalizable to any game in which avatars' movements are controlled by the players directly. 2) We study two novel dissimilarity measures between a pair of trajectories for robust trajectory representation and bot detection. 3) Based on real-life human traces, the performance evaluation results show that the scheme can achieve a detection accuracy of 98% or higher on a 700-s trace. As it is difficult to simulate human players' logic and determine how they control game characters, we believe that this approach has the potential to distinguish between human players and automated programs and thus merits further investigation.

The remainder of this paper is organized as follows. Section II contains a review of related works. In Section III, we introduce our case study game *Quake 2* and describe the game trace collection methodology. Section III-A details some basic observations about trace data belonging to bots and human users. In Section IV, to detect game bots, we consider two dissimilarities or differences between the trajectories of different types of players and combine them with a manifold learning methodology for data representation and classification. In Section V, we evaluate our approach's performance based on different dissimilarity measures and input traces of different length. Then, in Section VI, we summarize our conclusions.

II. Related Work

In recent years, a number of studies have employed machine learning techniques to detect or simulate bots in online games. For example, Yeung *et al.* [9] proposed using a dynamic Bayesian network (DBN) to model the aiming accuracy for *aimbot* detection in FPS games. In a DBN, the aiming accuracy depends on whether the player is cheating, whether the player or the target is moving, the aiming direction, and the distance between the player and the target. Since the possibility that a player is cheating is a random variable, the authors modeled it by a Markov chain. The model can detect cheaters with a high degree of accuracy, but it can only be applied to aimbots. Kim *et al.* [13] proposed detecting *auto programs* in MMORPGs based on the window events, which are generated by a player's key strokes, mouse clicks, and mouse movements. The events are collected during game play and used to compute statistics like the mean and standard deviation of the counts of certain events at regular intervals. Then, various classification schemes, such as the decision tree, the k-nearest neighbor (k-NN) classifier, the multilayer perceptron network, or the naive Bayesian classifier, are applied to determine whether automated programs are being used. Because of the high level of regularity exhibited by such programs, the window-event-based approach performs well irrespective of the classification method used.

Thurau *et al.* [14] attempted to create human-like game agents with machine learning approaches. They classified the behavior of human players into two categories: *perceptions* and *reactions*. The former includes a player's environmental information like the avatar's position and the distance between the avatar and nearby opponents; the latter includes a player's actions, such as the avatar's movement velocity and direction. Using the information from both categories, the authors created automatic human-like game agents. A number of learning approaches have been exploited in a series of papers by the same research group, including self-organizing maps [15], manifold learning [16], Bayesian networks [17], and waypoint maps [14], [18]. Equipped with these learning techniques, the proposed game agents can imitate human behavior very well compared to traditional rule-based game agents. However, the manifold learning approach in [16] only performs 3-D to 2-D mapping. We doubt that the *curse of dimensionality* (e.g., [10]) will arise in this case. Instead, in this work, based on avatars' movement trajectories, we apply a manifold learning framework with more than 200 original dimensions to detect the use of game bots.

A number of approaches for measuring the dissimilarities or differences between a pair of sequential traces have been proposed. For example, Keogh *et al.* [19] considered parameter-free descriptions of sequential data, while Pao *et al.* [20] studied the distance function between biological sequences. Both followed the work of Li *et al.* [21], who used the Kolmogorov complexity [22] to describe sequential data. Generally, the Kolmogorov complexity cannot be computed but some compres-

Fig. 1. Screen shot of *Quake 2*.

sion methods [20] can be used to approximate it.[2] The above methods only consider sequences of categorical values, so they cannot be applied directly to our problem, which takes inputs of numerical values. Lin *et al.* [23] proposed a symbolic representation called symbolic aggregate approximation (SAX) to deal with numerical valued time series. The key step of SAX involves discretizing continuous valued inputs to produce an approximate representation of the original inputs. The method has been applied successfully to many time series problems; however, in our case, it would be unnatural to use it to produce discretized data and compute the dissimilarity measures that are again in the continuous domain.

III. Data Description

In this section, we describe our case study game *Quake 2* and the procedures used to collect the game traces. We also analyze the navigation patterns in different traces.

1) Quake 2: *Quake 2* is a famous FPS game developed by id Software [24]. In FPS games, a player adopts the role of a particular character and shoots his enemies via the user interface shown in Fig. 1. Multiple players can participate in a game simultaneously, and they can cooperate to complete a mission. However, death-match games, in which each player tries to kill as many other participants as possible, are much more popular. *Quake 2* was nominated "The Best Game Ever" by *PC Gamer* in 1998, and went on to sell over one million copies [25]. One reason for the game's popularity is that it is easy to customize, and a large number of maps, player models, textures, and sound effects are available on the Internet. The game has been ported to many platforms other than PCs, for example, Nintendo 64, Playstation, Amiga PowerPC, and Xbox 360.

2) Human Traces: *Quake 2* supports a game-play recording function that keeps track of every action and movement, as well as the status of each character and item, throughout the game. With a recorded trace, one can reconstruct a game and review it from any position and angle desired with VCR-like operations. Players often use this function to assess their performance and combat strategies. Moreover, experienced players are encouraged to publish their game-play traces as teaching materials for novice gamers and thereby build a reputation in the gaming community.

TABLE I
Trace Summary

	Name	No	Length	Total	Active
1	Human	282	1000 seconds	78.0 hours	89%
2	CR	75	1000 seconds	20.8 hours	89%
3	Eraser	102	1000 seconds	28.3 hours	92%
4	ICE	60	1000 seconds	16.7 hours	67%

To ensure that our game traces represented the diversity of *Quake* players, we only used traces that players had contributed voluntarily. The traces were downloaded from the following archive sites: GotFrag Quake,[3] Planet Quake,[4] Demo Squad,[5] and Revilla Quake Site.[6] We mainly focus on the traces from the map called *The Edge*, one of the most well-known levels in death-match play. At this level, each player's sole goal is to kill as many other players as possible, until the time limit is reached. Traces on other maps called *The Frag Pipe* and *Warehouse* were also studied. However, the data sizes are relatively small and only presented in a supporting role.[7] In Section V-D, we study the detection power crossing different maps. As short traces contain little information, we only collected traces longer than 600 seconds.

3) Bot Traces: There are many game bots available for *Quake 2*. For this study, we selected three of the most popular bot programs for trace collection, namely CR Bot 1.14 [26], Eraser Bot 1.01 [27], and ICE Bot 1.0 [28].

To collect the game bot traces, we set up experiments on our own *Quake* server and ran a number of game bots to fight each other. The experiment setup was as follows.

1) In each game, 2–6 bots were selected at random to fight each other. Each session spanned 20 h.
2) The game trace was recorded at the server using the server record command.
3) The game's catch-the-flag mode was turned off, so the game bots continued fighting each other until the server shut down. The cheating mode was also disabled.
4) The AI levels of CR Bots and Eraser Bots were randomly set from 0 to 9 and 0 to 3, respectively.

We collected 1306 h of raw traces. Then, from each trace, we took the first 1000 s, the middle 1000 s, and another 1000 s near the end to compile our data set.[8] In total, we collected 143.8 h of trace data, as shown in Table I. The CR Bots, Eraser Bots, and all human players were active most of time ($\geq$89%). The ICE

[2]In fact, the Kolmogorov complexity and various definitions of entropy share similar properties. More details can be found in [12] and [22].

[3]http://www.gotfrag.com/quake/home/

[4]http://planetquake.gamespy.com/

[5]http://q2scene.net/ds/

[6]http://www.revilla.nildram.co.uk/demos-full.htm

[7]We show the result on The Edge map unless otherwise specified.

[8]We assume that the sections at the beginning, in the middle, and near the end of a trace are dissimilar, and can thus be considered as different samples. In this way, we can create more useful data items as input for our learning scheme; however, this preprocessing is not essential for our scheme to work properly.

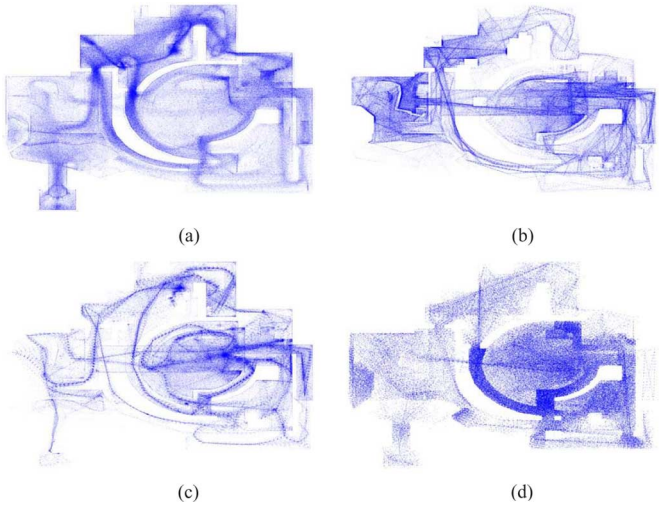


Fig. 2. Aggregated trajectories on The Edge map, for players belonging to the following groups: (a) human, (b) CR Bot, (c) Eraser Bot, and (d) ICE Bot. The figure shows that the routing of bot traces is more predictable than that of human traces, especially for the cases of CR Bot and Eraser Bot. Note that the bottom left-hand side corner of the CR Bot navigation map in (b) shows no bot presence. It may be difficult for some bots to visit the narrow area with a poor routing algorithm. We also have similar results for the trajectories obtained in The Frag Pipe map and Warehouse map (results not shown).

Bots were less active because they often remained idle in some places waiting for an opportunity to ambush other players.

A. Navigation Patterns and Preliminary Analysis

Next, we compare the avatar trajectories of human players and game bots based on certain observations. We consider the navigation patterns of different types of players. First, we analyze their aggregated navigation patterns and then, the patterns of individual trajectories.

1) Aggregated Navigation Patterns: We construct the aggregated navigation pattern of each player type by plotting all the observed coordinates in all traces of the particular player type on a map, as shown in Fig. 2. The high density areas in each figure are the places that players visit more frequently, while the sparse areas represent buildings, other types of obstacles that players cannot pass, or just areas that players are not interested in visiting. The figures show that the game level is formed by squares, plazas, and narrow alleys. This arrangement is designed specifically for death-match play, as the winding routes provide cover for players to hide, and the narrow alleys lead to intense fighting if players confront each other in these confined places. We observe that, even though all the movement traces were collected on the same map, the navigation patterns of different types of players are dissimilar. We summarize the differences below.

1) Human players tended to explore all areas on the map; thus, Fig. 2(a) shows the most complete terrain of the level. In contrast, the routing algorithms used by game bots may have had difficulty navigating certain places, so they never visited some parts of the map. For example, the bottom left-hand side corner of the CR Bot navigation map in Fig. 2(b) does not indicate any visits.
2) To reduce the probability of being attacked, human players normally avoid open spaces. Therefore, as shown in Fig. 2(a), human players avoided the plaza in the middle of the map, and stayed in the surrounding alleys instead. This is indicated by the high density of plots in the alleys. In contrast, game bots often stayed in the plaza, probably because it is a large space and it is easy to get everywhere from this area based on a simple routing algorithm.
3) Even though human players spent most of their time in narrow areas and confined spaces, there were large variations in their trajectories. There are two reasons for this phenomenon. a) The main routes are quite wide, so players move irregularly within the space rather than stay in the middle of a route. This may be due to players' preferences; hence, some players may move along the wall of the path, while others may walk straight, unless the avatar is blocked by a wall or other obstacles. b) As fights may occur anytime, anywhere, human players often move strategically to dodge current or potential attacks. In contrast, we find that the game bots adopt very different movement patterns over the routes. The movement paths of CR Bot and Eraser Bot [Fig. 2(b) and (c), respectively] are dense and easy to identify. This suggests that these bots tend to follow exact movement patterns when moving through the same alley. However, ICE Bot [Fig. 2(d)] exhibits a nearly uniform distribution over all possible points on the map. This implies that its routing algorithm decides the avatar's direction rather than its exact movement pattern, so that the probabilities of all points on the route are almost equivalent.

Clearly, the difference between the bots' routing patterns and those of human players explains the different aggregated patterns on the map.

2) Individual Trajectories: Having analyzed the aggregated navigation patterns of the different player types, we now examine their individual trajectories. We manually select a representative trace for each of the four player types (a human player plus three game bots). The avatar trajectory of each selected trace is shown in Fig. 3.

Even if we only observe one trace at a time, the difference between the player types is still apparent. Our observations about the aggregated navigation patterns still hold. Specifically, the narrower a place is, the higher the probability that human players will stay in that place, which is the opposite of the game bots' behavior patterns. Moreover, human players' trajectories contain much more irregularity and turns than those of bots. There are two possible explanations for this: 1) irregular moves reduce the chances of being attacked from behind; and 2) human decision making can be erratic and thus may not be logical all the time. A human player may change his/her mind any time and adjust the character's step and direction, basing the decision on unpredictable factors. In contrast, bots' trajectories are mostly characterized by straight and long paths.

The differences between the movement patterns of human players and game bots provide the conceptual framework for our trajectory-based behavior analysis and bot detection scheme. Even though bot developers may counter the detection algorithm by training bots to mimic human behavior, we argue that certain human behavior traits are difficult to emulate. While game bots' fixed movement patterns can be

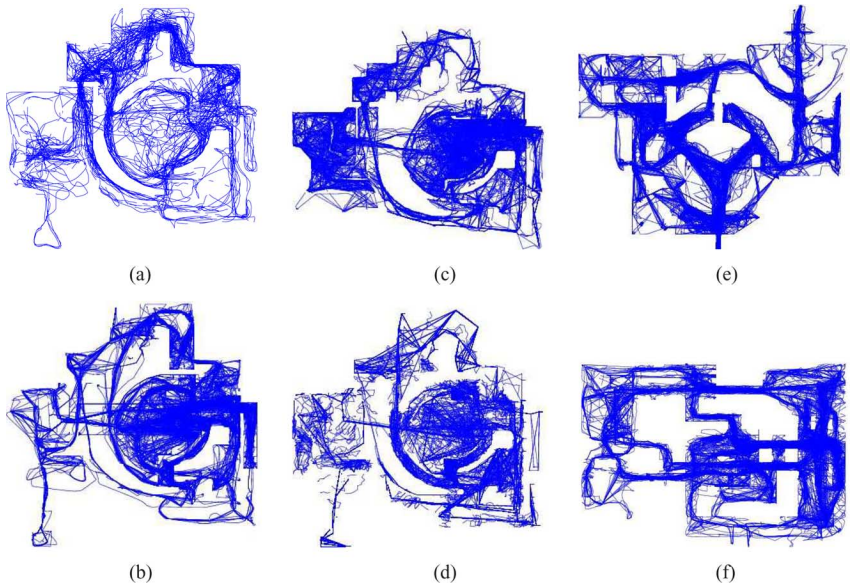


Fig. 3. (a)–(d) Game-play trajectories of a human player and three bot players in The Edge map; (e) and (f) game-play trajectories of an Eraser bot in The Frag Pipe map and in Warehouse map, respectively. The bots' trajectories exhibit less randomness/irregularity than the human's trajectory.

made more flexible by incorporating randomness into the navigation logic, evaluating whether a place is "dangerous" can be nontrivial. For example, human players tend not to stay in the central plaza on the map and thereby they reduce the probability of being attacked; however, it is difficult for game bots to "sense" the environment and "decide" to avoid staying in the plaza. Therefore, we believe that bot detection schemes based on avatar trajectories would be robust to bot developers' countermeasures. (This assumption is supported by the empirical study discussed in Section V-C.)

In summary, bot patterns are more regular than human patterns, and it is easier to predict whether a bot will go to a particular location than is the case with a human user. We use this observation as the basis for simple *discriminant analysis*. If we use a *binary* random variable X to describe an event where a trace $\mathbf{s}$ touches a location i within a certain period, we can compute the entropy of the random variable by

$$H_{\mathbf{s}i}(X) = -P(x)\log P(x) - (1-P(x))\log(1-P(x)).$$

The entropy values of human traces should be higher than those of bots, given a prespecified period. We test our conjecture on of 10 000-s traces[9] of *The Edge*, the map mentioned at the beginning of this section. Formally, we partition the original 2-D map into grids with a fixed size of 20 units, and count the number of times the trace $\mathbf{s}$ visits each grid. We then normalize the total number to between 0 and 1 as the distribution (divided by the number of steps taken by the avatar or 10 000 in this case), and compute the entropy of the distribution. In this way, we obtain the entropy of each location for each trace; and we can use the average entropy of a map as the discriminant to distinguish bots from human users. We use 80% of the traces as training input and compute the traces' average entropy values to set the decision threshold for different types of players. The remaining 20% of traces are used as test data to evaluate our conjecture. If the average entropy of a test trace is higher than the threshold, we label it as a human user; otherwise, we label it as a bot. As shown in Table II, the average entropy values are $H_{\text{human}} = 9.33$, $H_{\text{CR}} = 8.46$, $H_{\text{Eraser}} = 8.72$, and $H_{\text{ICE}} = 8.41$ for humans, CR Bots, Eraser Bots, and ICE Bots, respectively. We set the threshold at 8.95 to judge whether a trace is a bot or a human user. The method can achieve 88.57% test accuracy, as shown by the results in Table III. We need to emphasize that the detection based on entropy computation is sensitive to the length of input trajectories. We choose long enough traces (equal to 10 000 steps) so that the trace user starts to explore most of the

[9]We choose a trace longer than 1000 for better visualization effect and better performance based on entropy computation, which implies that the entropy computation is not as effective as the methods proposed in this paper.

TABLE II
SUMMARY OF HUMAN AND BOT ENTROPY VALUES. THE DECISION THRESHOLD BETWEEN HUMAN AND BOT PLAYERS IS SET AS THE MIDPOINT OF THE AVERAGE ENTROPY VALUES OF HUMAN AND BOT USERS; I.E., WE CHOOSE $(9.33 + 8.56)/2 = 8.95$ AS THE THRESHOLD

		Bot				
Class	Human	CR	Eraser	ICE	Total	Threshold
Entropy (The Edge map)	9.33	8.46	8.72	8.41	8.56	8.95

TABLE III
SUMMARY OF DATA AND RESULTS. THERE ARE 138 TRACES, EACH OF 10 000 S. WE USE 80% OF THEM FOR TRAINING AND THE REMAINDER FOR TESTING

	Data	Human	Bot			Error Rate
			CR	Eraser	ICE	
Numbers &	Training	75	20	27	16	6.52%
Results	Test	19	5	7	4	11.43%

space and we can see the discriminant results.[10] In Section IV, to further improve the detection power of our approach, with shorter trace inputs and higher detection accuracy, we propose a dissimilarity-based method for robust bot detection.

IV. Bot Detection Schemes

Our objective is to analyze the behavior patterns hidden in trajectories to distinguish between bots and human users. Based on the discussion in the previous section, we can simply use the average entropy as a feature, which yields a detection accuracy rate of 88.57%. In general, it is straightforward to consider several features of trace sequences that have been suggested by experts for bot detection. Chen *et al.* [29] recommended using various features, such as on/off activity, pace statistics, path statistics, and turn information, as the feature set for classification. They reported accuracy rates of 80%–90% for different combinations of the features. The above features, including average entropy, can also be combined to further enhance the classification performance.

However, expert knowledge is expensive and sometimes unreliable or biased. Generally, feature extraction is a difficult task if not an art. In this work, to detect bots from the trace inputs automatically, we employ two approaches for feature extraction and trajectory representation without the help of expert knowledge. The approaches try to measure the dissimilarity of pairwise trajectories, and the pairwise dissimilarities are used to find representatives in the new space. The representative points with specific *signatures* in the space are labeled as bots.

The input is a trajectory $\mathbf{s}$, or a series of location coordinates, in either a 2-D or 3-D space, i.e., $\mathbf{s} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_T)$ up to time T. Usually, T represents the effectiveness of the detection technique, or how quickly an alarm should be raised about a bot or a user who is cheating. The key step is to transform the trace data into a point in a new, probably low-dimensional space called the *representation space* and solve the detection problem in the space via a classification method.

We apply two dissimilarity measures and use a *manifold learning* technique called Isomap [11] to find the trace representation/embedding. Then, we adopt two methods, the k-NN algorithm and the support vector machine (SVM) model [30], [31], for classification in the representation space. Following convention, we treat bot traces as positive samples and human traces as negative samples to form a binary classification problem. Our bot detection algorithm comprises three parts: *dissimilarity measurement*, *trajectory representation*, and *bot detection via classification*. The first step measures the dissimilarities between pairs of trajectories. We utilize two measures, one without and one with temporal information. In the second step, Isomap is used to find the low-dimensional embeddings of trajectories given the pairwise dissimilarities. Then, given the embeddings in a low-dimensional space, the third step detects bots in the representation space. The generic algorithm is shown in Algorithm 1. In the following, we discuss the major components of the algorithm.

[10]The data sets from The Frag Pipe map and Warehouse map include many short trajectories from human users, therefore not appropriate to apply this entropy-based detection method. In general, the choice of entropy threshold to distinguish between human and bot users depends on the map layout. Different maps may induce or cause players to react differently. Just like the examples mentioned in the text, we can easily see different patterns from human and bot traces in open space, or narrow corridors of the map. Roughly speaking, if a map includes more of those regions, we expect to see larger gaps between the entropy values computed from human and bot traces.

TABLE IV
Notations

Notation	Definition
$\mathbf{s}_m(1:T)$	A trace sequence of length T, where $\mathbf{s}_m(1:T) = (\mathbf{x}_{m1}, \ldots, \mathbf{x}_{mT}), \quad m = 1, \ldots, M$. That is, there are total of M sequences.
λ_t	The step size $\|\mathbf{x}_{t+1} - \mathbf{x}_t\|$ at time t
θ_t	The angle between vector $\mathbf{x}_{t+1} - \mathbf{x}_t$ and the x-axis
$\Delta\lambda_t$	The step size change, i.e., $\Delta\lambda_t = \lambda_{t+1} - \lambda_t$
$\Delta\theta_t$	The angle change, i.e., $\Delta\theta_t = \theta_{t+1} - \theta_t$
$\mathcal{M}(\mu_\lambda, \mu_\theta, \sigma_\lambda, \sigma_\theta)$	The Markov chain model with the means μ_λ, μ_θ and the standard deviations σ_λ, σ_θ as the model's parameters, which decide the transitions of continuous random variables governed by two Gaussian models: μ_λ, σ_λ, the mean and the std for step size changes respectively, and μ_θ, σ_θ, the mean and the std for angle changes respectively.
$L(\mathbf{s}; \mu_\lambda, \mu_\theta, \sigma_\lambda, \sigma_\theta)$	Likelihood of sequence $\mathbf{s}$ given the model $\mathcal{M}(\mu_\lambda, \mu_\theta, \sigma_\lambda, \sigma_\theta)$
$\ell(\mathbf{s}; \mu_\lambda, \mu_\theta, \sigma_\lambda, \sigma_\theta)$	Log-likelihood with $\ell(\mathbf{s}; \mu_\lambda, \mu_\theta, \sigma_\lambda, \sigma_\theta) = \log L(\mathbf{s}; \mu_\lambda, \mu_\theta, \sigma_\lambda, \sigma_\theta)$
$\mathbf{D}^{SS}$	Based on the step-size distribution, the dissimilarity matrix records pairwise dissimilarities between each pair of traces
$\mathbf{D}^{MC}$	The dissimilarity matrix based on Markov chain modeling for pairs of traces

Algorithm 1: Generic algorithm for bot detection

Input: The new (unlabeled) trace seq. $\mathbf{s}_{M+1} := \mathbf{s}^*$ and a set of labeled traces $\{\mathbf{s}_1, \ldots, \mathbf{s}_M\}$,

Output: Label of $\mathbf{s}^*$

/* step 1: **Dissimilarity Measurement** */

1 **for** (m, m') *in* $\{1, \ldots, M+1\} \times \{1, \ldots, M+1\}$ **do**

2 | Compute dissimilarity $\mathbf{D}_{mm'} := d(\mathbf{s}_m, \mathbf{s}_{m'})$ [via (1) or (6)];

3 **end**

/* step 2: **Trajectory Representation** */

4 Given a matrix $\mathbf{D}$, apply Isomap to find embeddings of trajectories $\{\mathbf{s}_1, \ldots, \mathbf{s}_{M+1}\}$ in a low-dimensional space;

/* step 3: **Bot Detection via Classification** */

5 Given the low-dimensional embeddings, adopt classification method k-NN or smooth SVM (SSVM) for bot detection

Table IV summarizes the notations used in the remainder of the paper.

A. Dissimilarity Measurement

The first goal of our algorithm is to find a measure to describe the dissimilarity between two traces. We consider two cases.

1) No temporal information is considered. For instance, we can consider the trace as a set of steps $\mathbf{x}_{t+1} - \mathbf{x}_t$ without ordering.
2) We consider the whole trace, including the temporal information, to compute the pairwise dissimilarity.

1) Without Temporal Information: In this case, we consider the step information to detect bots. A step in a trace is the vector $\mathbf{x}_{t+1} - \mathbf{x}_t$, whose Euclidean step size is given by $\lambda_t = \|\mathbf{x}_{t+1} - \mathbf{x}_t\|$. We then estimate the distribution of the step size, i.e., the frequency counts of the step size after discretization.[11] The counts are collected in $B + 1$ bins as $(P_0, P_1, \ldots, P_B)$, based on the frequencies of the step sizes from 0 to a large number. We assume that a frequency count of step size 0 indicates a period of conversation, a rest period, hiding from intense fire, or waiting for the arrival of opponents. The resulting frequency vector can be used directly as input for our machine learning framework; i.e., it can be combined with classifiers for bot detection. However, in general, the performance is not satisfactory due to the high dimensionality of frequency vectors.

To avoid the *curse of dimensionality* [10], we adopt Isomap [11] as a dimension reduction technique to find low-dimensional embeddings of the frequency vectors and perform the classification task in the low-dimensional space. Isomap takes pairwise dissimilarity measures of input samples and outputs embeddings of those samples.

Given the step-size distributions, we compute their pairwise dissimilarity measure. In mathematics, given two quantized distributions, $P = (P_0, \ldots, P_b, \ldots, P_B)$ and $Q = (Q_0, \ldots, Q_b, \ldots, Q_B)$, it is natural to use the KL divergence [12]

$$d^{SS}(P,Q) = \sum_b P(b) \log \frac{P(b)}{Q(b)}$$

as the dissimilarity measure. In general, the KL divergence is not symmetric; however, we prefer a symmetric version

$$D^{SS}(P,Q) = d^{SS}(P,Q) + d^{SS}(Q,P). \tag{1}$$

For simplicity, we can also use the Euclidean metric[12]

$$D^E(P,Q) = \sqrt{\sum_b (P(b) - Q(b))^2}$$

to measure the distance between two data points/distributions. We use a symmetry matrix $\mathbf{D}^{SS}$ to store the dissimilarity measures (either D^{SS} or D^E) between pairs of distribution vectors obtained from two trajectories, where $\mathbf{D}^{SS}_{i,j}$ denotes the dissimilarity between trace i and trace j.

2) With Temporal Information: Alternatively, we can utilize the temporal information in the trace and employ the Markov chain model for dissimilarity measurement. Let $\mathcal{M}(\mu_\lambda, \mu_\theta, \sigma_\lambda, \sigma_\theta)$ denote the model of a trace sequence, where the transition parameters μ_λ and σ_λ describe the mean and the standard deviation of the step-size changes, respectively, and μ_θ and σ_θ describe the mean and the standard deviation of the angle changes, respectively.[13] Based on the Markovian property, between two step sizes λ_t and λ_{t+1} or coordinates in three consecutive time stamps $\mathbf{x}_t$, $\mathbf{x}_{t+1}$, and $\mathbf{x}_{t+2}$, we assume that

$$\begin{aligned} p(\Delta\lambda_t = \lambda_{t+1} - \lambda_t) &\sim \mathcal{N}(\mu_\lambda, \sigma_\lambda^2) \\ &= \frac{1}{\sqrt{2\pi}\sigma_\lambda} \exp\left(-\frac{(\Delta\lambda_t - \mu_\lambda)^2}{2\sigma_\lambda^2}\right) \end{aligned} \tag{2}$$

for step-size changes, and

$$\begin{aligned} p(\Delta\theta_t = \theta_{t+1} - \theta_t) &\sim \mathcal{N}(\mu_\theta, \sigma_\theta^2) \\ &= \frac{1}{\sqrt{2\pi}\sigma_\theta} \exp\left(-\frac{(\Delta\theta_t - \mu_\theta)^2}{2\sigma_\theta^2}\right) \end{aligned} \tag{3}$$

for angle changes.

We study a metric from information theory for measuring the dissimilarity between trajectories. In our design, each trace $\mathbf{s}$ has an associated model $\mathcal{M}(\mu_\lambda, \mu_\theta, \sigma_\lambda, \sigma_\theta)$ with a set of transition parameters μ_λ, μ_θ and σ_λ, σ_θ decided by the trace. Given a model $\mathcal{M}$, the log-likelihood $\ell(\mathbf{s}; \mathcal{M})$ of a trace $\mathbf{s}$ can be written as[14]

$$\begin{aligned} \ell(\mathbf{s}; \mathcal{M}) &= \log L(\mathbf{s}; \mathcal{M}) \\ &= \log\left(p(\mathbf{x}_1, \mathbf{x}_2) \prod_{t=1} p(\mathbf{x}_{t+2} | \mathbf{x}_t, \mathbf{x}_{t+1})\right) \\ &= \log p(\mathbf{x}_1, \mathbf{x}_2) + \sum_{t=1} \log p(\mathbf{x}_{t+2} | \mathbf{x}_t, \mathbf{x}_{t+1}) \end{aligned} \tag{4}$$

where L is the likelihood function and $p(\mathbf{x}_{t+2} \,|\, \mathbf{x}_t, \mathbf{x}_{t+1})$ is defined by

$$p(\mathbf{x}_{t+2} | \mathbf{x}_t, \mathbf{x}_{t+1}) = p(\Delta\lambda_t, \Delta\theta_t) = p(\Delta\lambda_t) p(\Delta\theta_t). \tag{5}$$

That is, given the coordinates of previous two steps $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$, we first generate $\Delta\lambda_t$ and $\Delta\theta_t$ based on (2) and (3), then we know λ_{t+1} and θ_{t+1} and we can decide $\mathbf{x}_{t+2}$ given the known step size and angle from time $t + 1$ to time $t + 2$. Given two traces $\mathbf{s}_i$ and $\mathbf{s}_j$ and their associated models $\mathcal{M}_i$ and $\mathcal{M}_j$, the distance[15] or dissimilarity between the traces will depend on how well one trace is described by the model for the other trace. First, given the model $\mathcal{M}$, we compute the code length of a trace $\mathbf{s}$ as a negative logarithm of the likelihood as

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

[11]In the measure, we count step-size frequencies according to several different predefined ranges called "bin," such as counting the number of step sizes from 0 to 1 length unit, from 1 to 2 length units, and so on. The whole set of step-size frequencies is also considered as a discretized version of the step-size distribution.

[12]This is also the case when we transform the data via Isomap to a low-dimensional Euclidean space, and the Euclidean metric is appropriate for such space.

[13]In real cases, μ_λ and μ_θ are very close to zero.

[14]We assume that the uniform initial distribution $P(\mathbf{x}_1, \mathbf{x}_2)$ and the probability can be ignored in the maximum likelihood computation.

[15]Note that we do not ensure triangle inequality in this case.

Note that $\mathcal{M}$ does not have to be the associated model of the trace $\mathbf{s}$. Therefore, we can define the distance between two trajectories $\mathbf{s}_1$ and $\mathbf{s}_2$ as

$$D^{MC}(\mathbf{s}_1, \mathbf{s}_2) = \frac{c(\mathbf{s}_1|\mathcal{M}_2) + c(\mathbf{s}_2|\mathcal{M}_1)}{c(\mathbf{s}_{12}|\mathcal{M}_{12})} \tag{6}$$

where $\mathbf{s}_{ij}$ is a new trace formed by concatenating the traces $\mathbf{s}_i$ and $\mathbf{s}_j$, and $\mathcal{M}_{ij}$ is the associated model of the concatenated[16] trace $\mathbf{s}_{ij}$. In this way, for each pair of traces, we derive pairwise distance values, which will be used as input to find the representation of the traces. The dissimilarity values of pairwise traces are stored in a symmetric matrix $\mathbf{D}^{MC}$. The matrix $\mathbf{D}^{SS}$ or $\mathbf{D}^{MC}$ will in turn be input to Isomap in order to find the representations/embeddings of the trajectories. We discuss the results of choosing different dissimilarity measures in Section V.

Parameter Estimation for the Markov Chain: We assume that the trace sequence $(\mathbf{x}_1, \mathbf{x}_2, \ldots)$ for the step-size changes and angle changes follows the Markovian property. As mentioned earlier, we use a Gaussian-distributed transition as the transition function to approximate the step-size changes and angle changes, which are centered at μ_λ and μ_θ and with variances σ_λ^2 and σ_θ^2, respectively. The parameters can be estimated directly from the sample means and variances of the related data inputs. Given a trace, we can compute the differences between consecutive step sizes as $\Delta\lambda_t = \lambda_{t+1} - \lambda_t$ and estimate the mean and the variance via the sample mean $\hat{\mu}_\lambda$ and the variance $\hat{\sigma}_\lambda^2$. Similarly, for the angle changes, we can estimate μ_θ and σ_θ^2 via the sample mean $\hat{\mu}_\theta$ and the variance $\hat{\sigma}_\theta^2$.

B. Trajectory Representation

Trajectory representation seeks to represent a set of trajectories in a Euclidean space such that the Euclidean distance in the space fully represents the relations between the trajectories. In this study, we consider that two traces are similar if:

1) they both have small measurements in (1) or (6); or,
2) they are both similar to a third trace.

The second criterion means that two trajectories $\mathbf{s}_1$ and $\mathbf{s}_2$ are friends if they have a common friend $\mathbf{s}_3$, even if they do not have small values of D^{SS} or D^{MC} themselves. To find a metric to satisfy these criteria, we adopt Isomap [11] as the representation technique. The rationale behind this choice is that there is a high degree of variance among the trace sequences of human users and bots; therefore, it is difficult to propose a universally effective rule for detecting bots or identifying particular behavior patterns from trace sequences. With the second criterion, even if two trajectories are not highly similar, we can deem them close to each other simply because they are both similar to a third trajectory. A friendship that has such a transitive property can help us determine the *global* distance between pairwise trajectories.

The goal of Isomap is to find a representation in an intrinsic space in which it tries to maintain the neighborhood relationship between each pair of trajectories locally; however, globally, a geodesic distance between the two points/trajectories is substituted to describe their distance/dissimilarity. Given a dissimilarity matrix $\mathbf{D}$ ($\mathbf{D}^{SS}$ or $\mathbf{D}^{MC}$ in our case), the Isomap process can be divided into three steps. 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. 3) Take the pairwise (geodesic) distance as the input and apply multidimensional scaling (MDS) to find the global Euclidean coordinates of the points. The "optimal" dimensionality for separating the different kinds of trajectories can be estimated by finding the "elbow" point in the residual variance curve [11].

Figs. 4 and 5 show the results of applying Isomap[17] given the dissimilarity measures derived from the step size (without temporal information) and from the Markov chain model (with temporal information), respectively. The (green) circles indicate the traces of human users, while the others are obtained from several different bots. Among them, CR Bots (the cross symbols) and the human players appear to have the highest variances, but the ICE Bots exhibit relatively low variances. More importantly, data items with different labels are well separated. However, such discriminative results cannot be obtained if we use the well-known principal component analysis (PCA) method [33] for dimension reduction, as shown in Fig. 6. It is noteworthy that the representation derived by the Markov-chain-based measure is visually better than the one derived by the step-size measure because it includes temporal information about the trace. In Fig. 5, points/trajectories of the same type are clustered together, but that is not the case in Fig. 4. We believe that adding temporal information provides a better representation of trajectory behavior. Moreover, as we will show later, the classification in the representation space derived by the Markov chain model is more accurate than the one derived by the step size only. After Isomap finds a low-dimensional representation of the data, we can use any classification scheme, e.g., the k-NN algorithm or SVM, to label a new trace (i.e., either a bot or a human player).

C. Bot Detection via Classification

Given the trajectory representation, in principle, we can use any classification or clustering method for bot detection. In this study, we adopt SSVM, which tries to solve an unconstrained minimization problem [34], and the k-NN algorithm for most of our evaluations. We assume that the trajectory representations $\mathbf{z}_m$ are located in an N-dimensional space. Their associated labels are denoted by y_m.

1) k-Nearest Neighbors: The k-NN algorithm is one of the oldest and most intuitive classification methods, and many applications demonstrate its competitive performance compared to other classifiers (e.g., [35]). Under k-NN, the class label of a new trace is decided by the class labels of the traces surrounding it. One of the keys to the successful application of k-NN is the

[16]We can treat $\mathbf{s}_{12}$ and $\mathbf{s}_{21}$ as virtually the same to generate similar models between $\mathcal{M}_{12}$ or $\mathcal{M}_{21}$. The only factor that makes a difference is the transition at the concatenation point between $\mathbf{s}_1$ and $\mathbf{s}_2$.

[17]We only present data in 2-D for visualization purposes. In general, the detection or classification task is executed in the space of intrinsic dimensionality. The embedding produced by Isomap often suggests some meaningful insight for the matter of understanding patterns from humans or bots, if the pattern can be visualized in low-dimensional space. For instance, in this scenario, an axis may indicate that the avatars turn smoothly or abruptly; or tend to go in a constant step size or in a varied step size. Unfortunately, according to the traces superimposed on the Isomap results, due to the difficulty of visualizing the trace data, it is not easy to find out what the axes mean in this scenario.

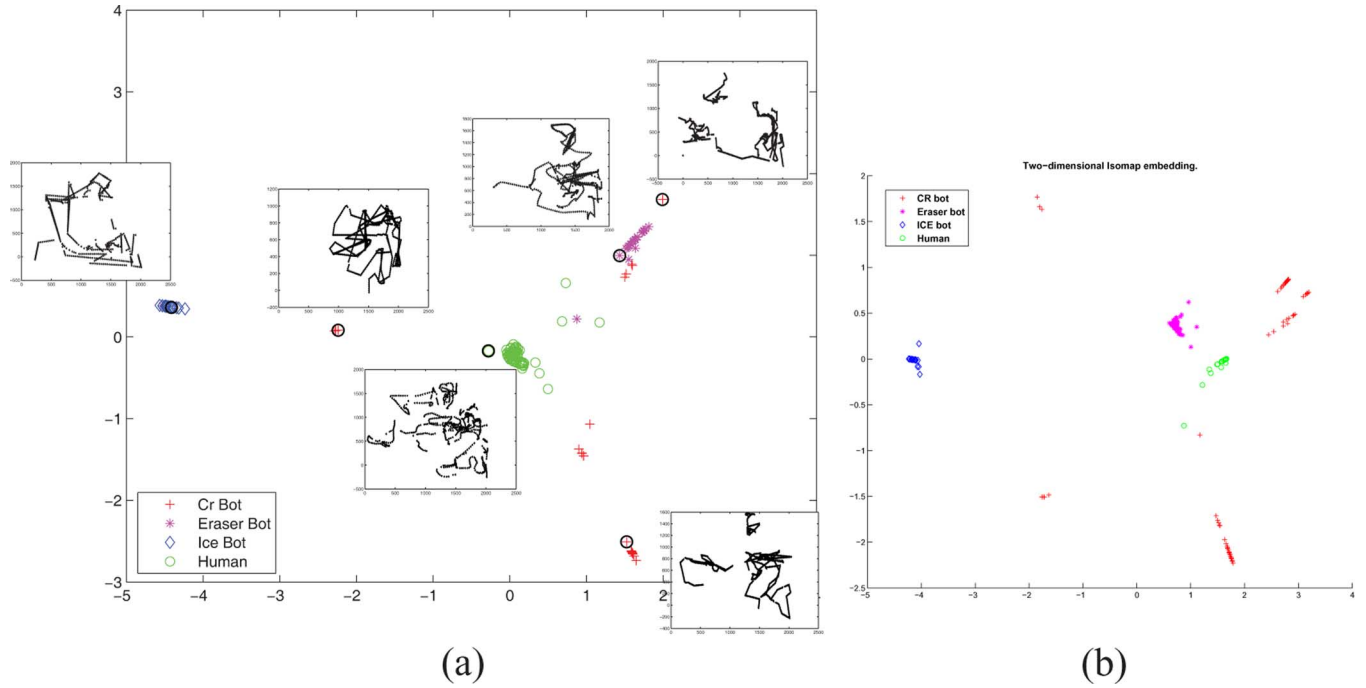


Fig. 4. Based on step-size distribution, the trace representation (and the associated trace superimposed on the side) by Isomap, for traces from (a) The Edge map and (b) The Frag Pipe map where a point represents a trace of a human user (green circle) or from a bot (others). The x- and y-axes are the first and second principal coordinates [32] from Isomap. As the figure shows, the human data and the bot data are well separated. Classification in this space using k-NN or SVM (or SSVM) can be performed with a high degree of accuracy.

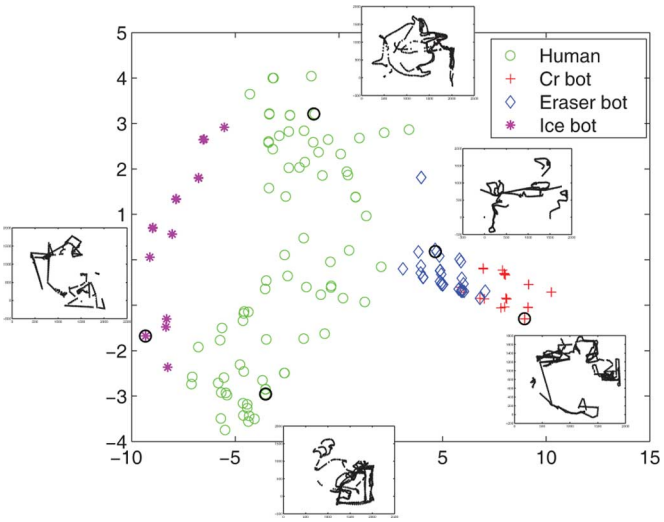


Fig. 5. Based on Markov chain model and code length computation, the trace representation (and the associated trace superimposed on the side) after Isomap, where a point represents a trace of a human user (green circle) or from a bot (others). The x- and y-axes are the first and second principal coordinates from Isomap. Compared to Fig. 4, it seems that the traces from different player types have well-clustered groups by the Markov-chain-based dissimilarity measure (with temporal information) rather than the dissimilarity measure based on step size (without temporal information).

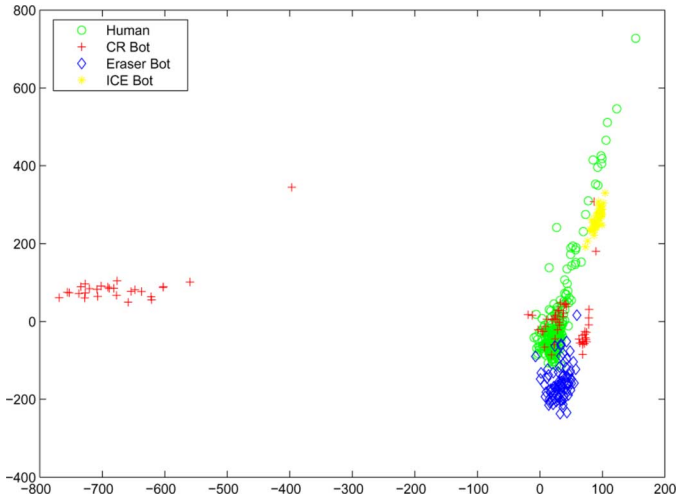


Fig. 6. Based on step-size distribution, the data representation by PCA, where the x- and y-axes represent the first and second principal components, respectively. The points of human users and bots overlap and they are not distinguishable from each other.

choice of an appropriate metric. For instance, using KL divergence in the original space of the step-size distribution is not as effective as working on the representation space found by Isomap.

2) Support Vector Machines: SVMs are well suited for solving binary classification problems like bot detection. Theoretically, in a linear case, by selecting the separating hyperplane $\mathbf{w}^T\mathbf{z}+\mathbf{b}=0$ that maximizes the margin between positive and negative samples, we can obtain a classifier that minimizes the *generalization error* [30], [31]. Specifically, finding an optimal classifier is equivalent to minimizing a functional composed of the training error term and the regularization term as follows:

$$\begin{aligned} \min_{(\mathbf{w},\mathbf{b},\xi)\in R^{N+1+M}} \quad & C\sum_{m=1}^{M}\xi_m+\frac{1}{2}\|\mathbf{w}\|_2^2 \\ \text{s.t.} \quad & y_m(\mathbf{w}^T\mathbf{z}_m+\mathbf{b})+\xi_m\geq 1 \\ & \xi_m\geq 0, \qquad \text{for } m=1,2,\ldots,M \end{aligned}$$

where ξ_m denotes the positive slack variables, and C is a positive parameter that controls the balance between the training error and the margin maximization term. In a nonlinear case, the kernel trick [36] can help us find a nonlinear separating surface between positive and negative samples. Several variants of the typical SVM model have been proposed; for instance, the SSVM, which tries to solve an unconstrained minimization problem instead [34]. In this study, we use SSVM to evaluate our framework. We consider both linear and nonlinear versions.

V. Performance Evaluation

Our experiment is composed of four parts. The first part evaluates different bot detectors in terms of the error rate, false positive rate, and false negative rate, and demonstrates the effectiveness of our proposed methods. The second part assesses the detection rates based on inputs of different length. In the third part, we study the scenario where bot users try to counter a bot detection algorithm by mimicking human behavior. We add some noise to the bot input so that it looks similar to human input. The experiment shows that our detectors are still robust in this situation. At last, we discuss the problem of bot detection crossing different maps. Based on our result, the detection accuracies have at most 1% difference between the detection in a single map and the detection consisting of many maps.

To evaluate the performance, we use the detection error rate, which is measured by a tenfold cross-validation procedure. In other words, the whole data set is partitioned into ten subsets of more or less equal size, with stratification.[18] Then, nine of the subsets are used for training and the tenth is reserved for testing. The procedure is repeated ten times using different partitions to obtain an average result. Based on the data set described in Table I, there are 519 data items, of which 237 are positive samples (bots) and 282 are negative samples (human). As mentioned earlier, the length of each sample is 1000 s in all our experiments, unless otherwise specified (see Section V-B). Before discussing the experiment results, we define the parameters used in our evaluation.

a) Dissimilarity Measure Without Temporal Information: For the measure that takes the step size as the input, the training set is compiled by transforming each trace into a distribution of step sizes, as described in Section IV. The distribution is discretized and partitioned into $B + 1 = 201$ bins for each trace,[19] where the bin height is the value of the probability mass function or frequency counts; therefore, a data item is in a 201-dimensional space. To apply the Isomap procedure, the neighborhood graph is defined by considering the k-NN ($k_1 = 5$) of each sample. Then, after dimension reduction to a low-dimensional representation space (of dimensionality equal to five), we use an SSVM, or simply k-NN for classification (setting $k_2 = 13$ in the representation space). In the original space, the number of data items considered as neighbors should be limited ($k_1 = 5$ as mentioned above) due to the possible curse of dimensionality; however, this can be relaxed to a larger number ($k_2 = 13$ as mentioned) in a dimension-reduced space.

b) Dissimilarity Measure With Temporal Information: To consider the temporal information, we input the whole trajectory to a (first-degree) Markov chain model. Given two trajectories and their associated models, we use each model alternately to describe the other model's trajectory to find their dissimilarity. An SSVM or k-NN with $k_2 = 13$ is adopted for the classification. We discuss the performance of each approach in Sections V-B–V-E.

A. Effectiveness of the Proposed Methods

Next, we evaluate our two dissimilarity measures by combining with the k-NN classifier and SSVM classifier, for bot detection. We consider the following bot detection schemes: (1k) k-NN with KL divergence as the metric, given the frequency vector of the step size; (1s) SSVM,[20] also given the frequency vector of the step size; (2k) k-NN, applied in the low-dimensional space, which is derived from Isomap with the step-size input; (2s) Isomap followed by SSVM, with the step-size input; (3k) Isomap followed by k-NN, where the input consists of the pairwise dissimilarities based on the code length described by the Markov chain; and (3s) Isomap followed by SSVM, also with Markov chain modeling. Note that Isomap is only used to find the representation space in the series (2x) (i.e., 2k or 2s) and (3x). Series (2x) is based on the dissimilarity measure derived from the step-size inputs, i.e., without temporal information; on the other hand, series (3x) is based on the Markov-chain-based dissimilarity measure, i.e., with temporal information. The k-NN algorithm is considered a naive classification method, whereas SSVM is deemed a sophisticated method.

Our experiment results demonstrate that, in terms of performance, k-NN combined with Isomap (manifold learning) is comparable to any other methods, while k-NN is very efficient in terms of time complexity compared to SVM (or SSVM). Second, the approach with Isomap performs better than the one without Isomap. Third, the approach that considers temporal information, i.e., the input with the code length derived by the Markov chain model, outperforms all the other methods.

Tables V and VI show the performances of several bot detection methods. Both k-NN and SSVM are applied with and without Isomap; and we use the linear and nonlinear versions of SSVM. Most of the classification methods yield error rates[21] of less than 2%; among them, Isomap combined with nonlinear SSVM achieves perfect classification results for inputs with and without temporal information. Overall, the methods that employ Isomap yield better results than the methods that do not use it. Moreover, the methods based on the input with temporal information outperform those without temporal information in the input. Finally, the methods based on SSVM outperform those based on k-NN.

[18] In other words, the set is divided into several groups that contain similar percentages of positive and negative samples.

[19] That is, in the discrete case, the bins record the numbers of steps with length from 0 to 1, from 1 to 2, and so on.

[20] As mentioned previously, we adopt SSVM instead of SVM because it achieves a better performance.

[21] As mentioned earlier, Isomap combined with linear SSVM may not be effective because the decision boundary tends to be nonlinear in a low-dimensional space. For ease of visualization, we do not show the result of Isomap combined with linear SSVM in subsequent graphs because it is not comparable to the results of the other methods.

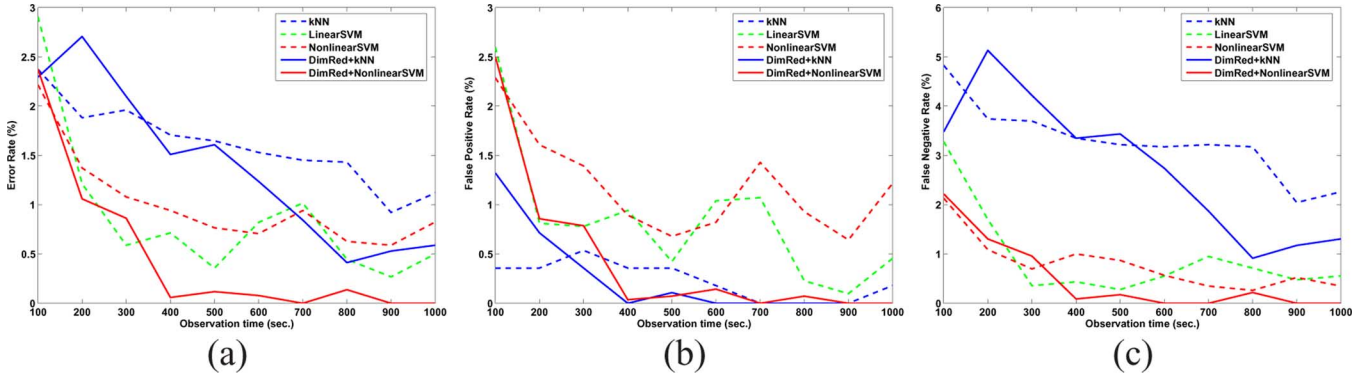


Fig. 7. Results obtained by six detection schemes based on the step-size input, given trajectories of different length: (a) error rates, (b) false positive rates, and (c) false negative rates (compared to Table V). The results are similar to those reported in Section V-A: 1) the methods combined with Isomap outperform those without it; and 2) the SSVM-based methods outperform those based on k-NN, except for the false positive results.

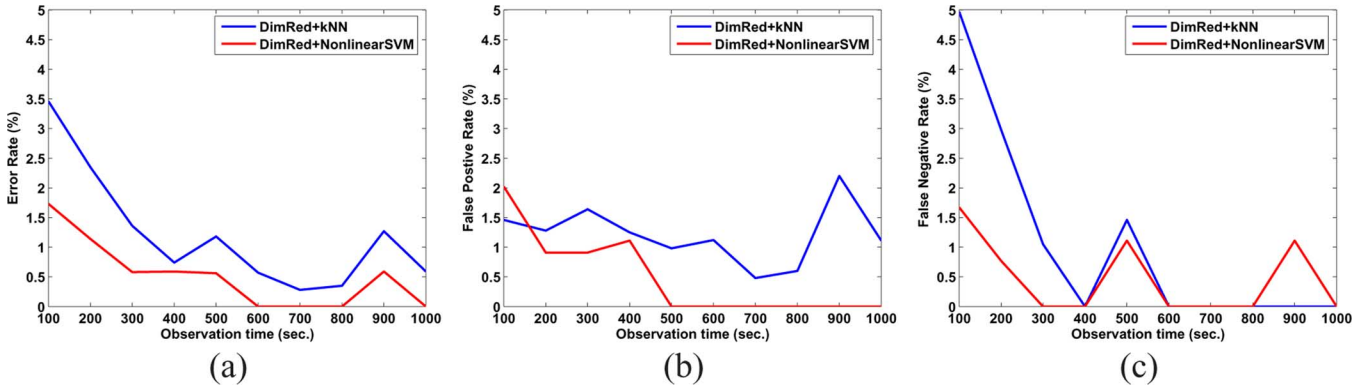


Fig. 8. Given different lengths of trajectories, the results obtained by two detection schemes, based on the Markov-chain-described code length: (a) error rates, (b) false positive rates, and (c) false negative rates (compared to Table VI). The results are similar to those in Fig. 7: 1) the methods combined with Isomap outperform those without it; and 2) the SSVM-based methods outperform those based on k-NN. Moreover, the methods that consider temporal information (using the Markov chain model) are more effective than those that do not include the information (i.e., they use the step-size input).

TABLE V
THE RESULTS BASED ON SIX DIFFERENT DETECTION SCHEMES, GIVEN THE STEP-SIZE INPUTS, I.E., NO TEMPORAL INFORMATION ARE CONSIDERED. THE AVERAGE RESULTS OF THE FALSE POSITIVE RATES, FALSE NEGATIVE RATES, AND ERROR RATES, AFTER TEN REPEATS OF THE TENFOLD CROSS-VALIDATION PROCEDURE. OVERALL, THE PERFORMANCE IS ENHANCED IF ISOMAP IS APPLIED. THE ONLY EXCEPTION IS THE POOR PERFORMANCE WHEN APPLYING LINEAR SSVM IN A LOW-DIMENSIONAL SPACE. THE DECISION BOUNDARY TENDS TO BECOME NONLINEAR IN A LOW-DIMENSIONAL SPACE; THUS, LINEAR SSVM MAY NOT BE APPROPRIATE IN THIS CASE

Classification Methods	FP(%) / FN(%)	Err(%)
(1K) kNN	0.00 / 3.22	1.45
(1SL) LINEAR SSVM	1.07 / 0.95	1.02
(1SN) NONLINEAR SSVM	1.43 / 0.35	0.94
(2K) ISOMAP + kNN	0.00 / 1.30	1.16
(2SL) ISOMAP + LINEAR SSVM	0.00 / 25.34	11.43
(2SN) ISOMAP + NONLINEAR SSVM	0.00 / 0.00	0.00

TABLE VI
THE RESULTS BASED ON THREE DIFFERENT DETECTION SCHEMES GIVEN THE INPUTS OF THE MARKOV CHAIN-DESCRIBED CODE LENGTH, I.E., WITH TEMPORAL INFORMATION. THE AVERAGE RESULTS OF THE FALSE POSITIVE RATES, FALSE NEGATIVE RATES, AND ERROR RATES, AFTER TEN REPEATS OF THE TENFOLD CROSS VALIDATION. COMPARED TO, ONCE AGAIN, WE FIND THAT THE PERFORMANCE FOR SERIES (1X) IS ENHANCED BY APPLYING ISOMAP. MOREOVER, WITH TEMPORAL INFORMATION INCLUDED, THE PERFORMANCE IS BETTER THAN THAT OF THE CLASSIFICATION RESULT BASED ON THE STEP-SIZE INPUTS (NO TEMPORAL INFORMATION)

Classification Methods	FP(%) / FN(%)	Err(%)
(3K) ISOMAP + kNN	1.11 / 0.00	0.59
(3SL) ISOMAP + LINEAR SSVM	2.67 / 0.91	2.32
(3SN) ISOMAP + NONLINEAR SSVM	0.00 / 0.00	0.00

B. Using Trajectories of Different Length

Since we want to detect bot users as early as possible, we can analyze the performance when only a partial input trace is given, rather than wait for a whole input sequence. As shown in Figs. 7 and 8, one method may be superior to another for inputs of different lengths, but the results are similar to those reported in the previous section.

1) The methods that use Isomap outperform those that do not use it.
2) The methods that consider temporal information (Markov chain model) outperform those that do not include it (step-size input).
3) The SSVM-based methods outperform the k-NN-based methods.

C. With Noise

The k-NN and SSVM methods combined with Isomap outperform those without it, even if noise is added to the bot trajectories (a common camouflage strategy) to counter the detection algorithm.

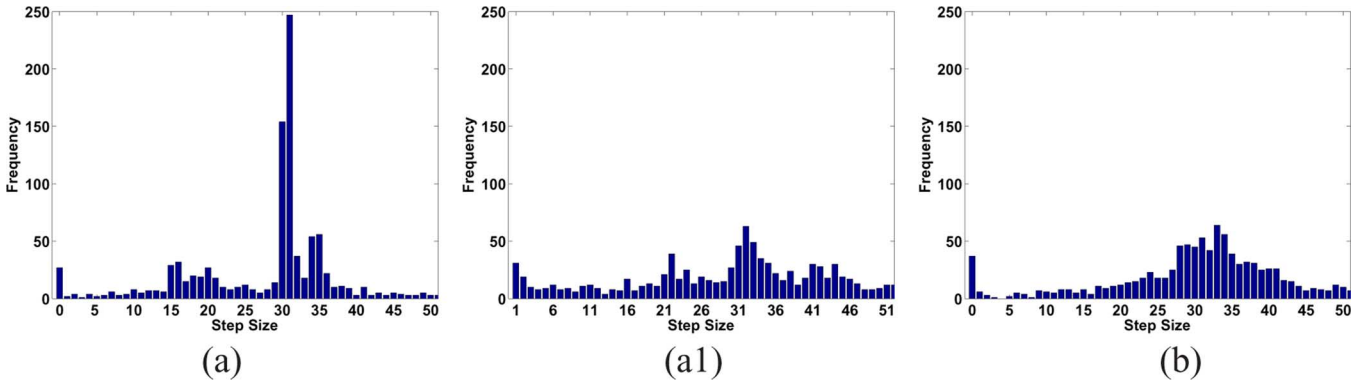


Fig. 9. Histograms of the step sizes for trajectories of (a) bot users and (b) human users. A large number of constant size steps are found in bot trajectories, but not in human trajectories. However, after adding some Gaussian noise in (a) to (a1), we obtain a histogram that is hard to distinguish, at least visually, from that for a human user.

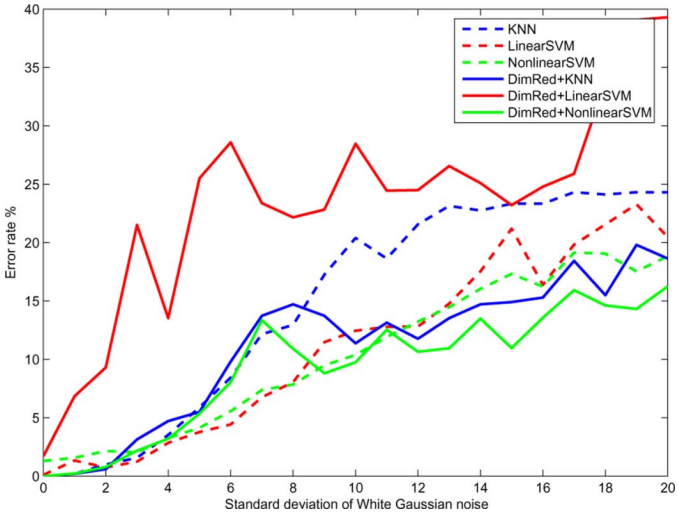


Fig. 10. Some white noise is added to the trajectories to test the robustness of the detection methods based on step-size input. The x-axis shows the standard deviation going higher in the rightward direction. Mostly, the methods combined with Isomap perform better than those methods without it.

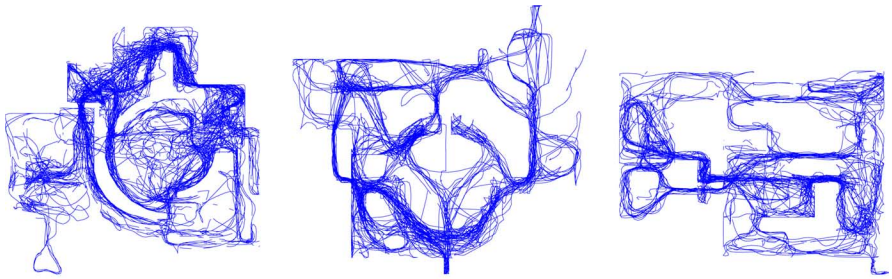
The step sizes of common bots have very regular distributions, as shown in Fig. 9(a), which is the histogram of the step size derived from a CR Bot's trajectory. That is, a bot tends to maintain a constant step size (around 32 in this case), which is not usually observed in human trajectories. Such features could easily be identified by a smart detector. Therefore, it is understandable that a bot user will try to avoid detection by adding some *white noise* to the step size. Our detector can deal with this kind of camouflage. Fig. 10 shows the results when different levels of white noise are added to the bot's trajectory in the step-size domain. After adding the noise, the distribution may not be visually distinguishable from a regular human trajectory [as shown by comparing Fig. 9(a1) and (b)]. Nevertheless, the experiment shows that our method can detect bot users via the step-size dissimilarity measure with a very low error rate. Once again, in most cases, the methods with Isomap perform better than those without it. Moreover, in terms of accuracy, SSVM-based approaches usually outperform k-NN-based approaches in terms of accuracy.

When temporal information is considered, the dissimilarity measure based on the Markov chain model is even more effective than the approach that only considers step-size inputs. With input trajectories equal or longer than 1000 s, all bots are detected with 100% accuracy.

D. Crossing Different Maps

Sometimes human movement may be restricted by the environment around him/her. For example, in a game, imagining that we are in a tunnel, then we can only move forward or backward in such a condition, and both of our flanks are suffocated by the surroundings. We would like to ensure that a model built for one map can be used for another map. We proceed to study this problem.

In this section, we would like to test the influence of the traces from different maps to our framework. We use traces from three maps which are very dissimilar to each other. The collected data information is in Table VII. The map The Edge is the simplest, containing some plazas and tunnels. The map The Frag Pipe contains many tunnels, and if the players encounter others, they have no cover and their movement is highly restricted by the surroundings we mentioned before. The map Warehouse has a very complicated structure and contains many floors. In this map, players can easily get lost in the map due to the complex landform. Three traces collected from the three maps are shown in Fig. 11 for visualization purpose. In the experiments, we collect traces from different maps together for the tenfold cross validation, and the validation test is carried out without acknowledging the source of the trace. As the experiment results show in Fig. 12, the performance is still good and not much affected by the cross-map effect. The accuracy from the model trained by traces from different maps has a difference of up to 1%, from the accuracy from the model built for a single map, and the performance of the approach combining Isomap and nonlinear SVM remains the best compared to other approaches. We should also emphasize that our model is based on features computed from local movements, such as step size, step-size changes, and angle chances. Those statistics are less likely to be influenced by map layout, compared to the statistics based on long-term movements. Therefore, in terms of keeping similar performance across different maps, the proposed method is



(a) (b) (c)

Fig. 11. Human traces from three maps. Obviously, the terrain of the three maps is totally different. The Edge has the simplest landform which is particularly designed for novice. The Frag Pipe has many tunnels. Warehouse has a complicated landform and players can easily get lost in the map.

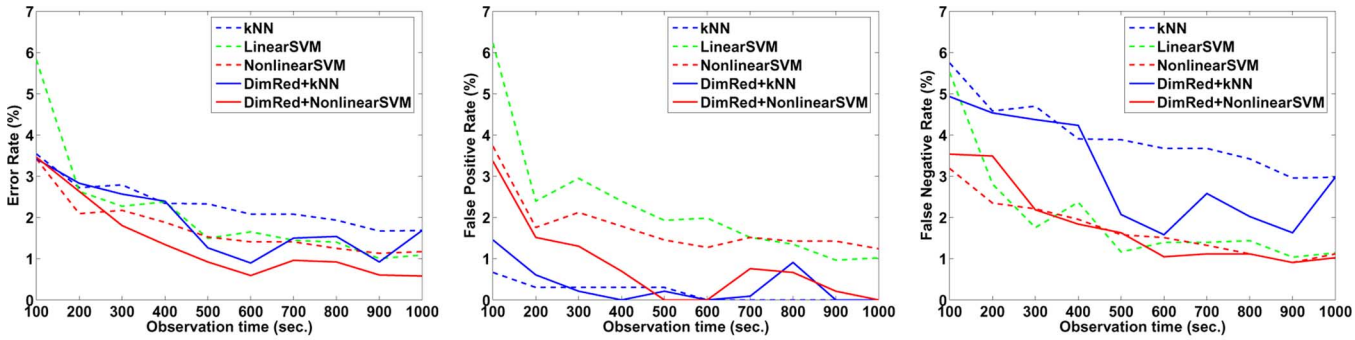


Fig. 12. Results crossing different maps, by six detection schemes based on the step-size input, given trajectories of different length: (a) error rates, (b) false positive rates, and (c) false negative rates. The results are similar to previous results, about 1% difference between them at most (compared to Fig. 7).

TABLE VII
DATA SUMMARY FOR CROSS-MAP EXPERIMENT

Map Name	Number of Traces			
	Human	Bot		
		CR	Eraser	ICE
The Edge	282	75	102	60
The Frag Pipe	24	69	18	42
Warehouse	24	45	27	0

favored over other methods that need features from global or long-term movements.

VI. CONCLUSION

We have proposed a trajectory-based approach for detecting game bots. Specifically, we employ Isomap to find an appropriate space for trajectory representation, and then use k-NN or SSVM to perform supervised classification. The evaluation results demonstrate that, based on real-life *Quake 2* traces, our approach can achieve a detection accuracy of 98% or higher on a 700-s trace. We believe that it is generally difficult to simulate human players' behavior when they are controlling game characters. Humans may not be totally logical; however, most of the time they obey rules or follow strategies that may not be easily sensed by bot users or automated programs. To some extent, the results of our study support this conjecture. The experiment results show that the proposed method can distinguish between human players and automated programs. Thus, we believe that the method merits further investigation by the game community.

ACKNOWLEDGMENT

The authors would like to thank anonymous referees for their constructive criticisms.

REFERENCES

[1] K.-T. Chen, J.-W. Jiang, P. Huang, H.-H. Chu, C.-L. Lei, and W.-C. Chen, "Identifying MMORPG bots: A traffic analysis approach," *EURASIP J. Adv. Signal Process.*, vol. 2009, 2009, DOI: 10.1155/2009/797159, article id 797159.

[2] P. Golle and N. Ducheneaut, "Preventing bots from playing online games," *Comput. Entertain.*, vol. 3, no. 3, p. 3, 2005.

[3] J. Dibbell and M. Video, "The life of the Chinese gold farmer," *The New York Times*, Jun. 17, 2007 [Online]. Available: http://www.nytimes.com/2007/06/17/magazine/17lootfarmers-t.html?ex=1339732800&en=a6282d1ddf608fc1&ei=5088&partner=rssnyt&emc=rss

[4] K.-T. Chen and L.-W. Hong, "User identification based on game-play activity patterns," in *Proc. 6th ACM SIGCOMM Workshop Netw. Syst. Support Games*, 2007, pp. 7–12.

[5] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford, "CAPTCHA: Using hard AI problems for security," in *Proc. Eurocrypt*, 2003, pp. 294–311.

[6] P. Hingston, "A Turing test for computer game bots," *IEEE Trans. Comput. Intell. AI Games*, vol. 1, no. 3, pp. 169–186, Sep. 2009.

[7] T. P. Novak, D. L. Hoffman, and A. Duhachek, "The influence of goal-directed and experiential activities on online flow experiences," *J. Consumer Psychol.*, vol. 13, no. 1, pp. 3–16, 2003.

[8] S. Ila, D. Mizerski, and D. Lam, "Comparing the effect of habit in the online game play of Australian and Indonesian gamers," in *Proc. Australia New Zealand Marketing Assoc. Conf.*, 2003.

[9] S. F. Yeung, J. C. S. Lui, J. Liu, and J. Yan, "Detecting cheaters for multiplayer games: Theory, design and implementation," *Consumer Commun. Netw. Conf.*, vol. 2, pp. 1178–1182, 2006.

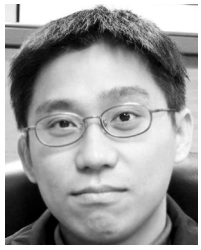
[10] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer-Verlag, 2006.

[11] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, Dec. 2000.

[12] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. New York: Wiley-Interscience, 2006.
[13] H. Kim, S. Hong, and J. Kim, "Detection of auto programs for MMORPGS," in *Proc. Adv. Artif. Intell.*, 2005, pp. 1281–1284.
[14] C. Thurau, C. Bauckhage, and G. Sagerer, "Learning human-like movement behavior for computer games," in *Proc. 8th Int. Conf. Simul. Adaptive Behavior*, 2004, pp. 315–323.
[15] C. Thurau, C. Bauckhauge, and G. Sagerer, "Combining self organizing maps and multilayer perceptrons to learn bot-behavior for a commercial game," in *Proc. GAME-ON Conf.*, 2003, pp. 119–123.
[16] C. Thurau and C. Bauckhage, "Towards manifold learning for gamebot behavior modeling," in *Proc. Int. Conf. Adv. Comput. Entertain. Technol.*, 2005, pp. 446–449.
[17] C. Thurau, T. Paczian, and C. Bauckhage, "Is Bayesian imitation learning the route to believable gamebots?," in *Proc. GAME-ON North America*, 2005, pp. 3–9.
[18] C. Thurau and C. Bauckhage, M. Merabti, N. Lee, and M. Overmars, Eds., "Tactical waypoint maps: Towards imitating tactics in FPS games," in *Proc. 3rd Int. Game Design Technol. Workshop Conf.*, 2005, pp. 140–144.
[19] E. Keogh, S. Lonardi, and C. A. Ratanamahatana, "Towards parameter-free data mining," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Disc. Data Mining*, 2004, pp. 206–215.
[20] H.-K. Pao and J. Case, "Computing entropy for ortholog detection," in *Proc. Int. Conf. Comput. Intell.*, 2004, pp. 89–92.
[21] 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*, vol. 17, no. 2, pp. 149–154, 2001.
[22] M. Li and P. Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications*, 2nd ed. New York: Springer-Verlag, 1997.
[23] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proc. 8th ACM SIGMOD Workshop Res. Issues Data Mining Knowl. Disc.*, 2003, pp. 2–11.
[24] [Online]. Available: http://www.idsoftware.com/
[25] Id Software: Id History [Online]. Available: http://www.idsoftware.com/business/history/
[26] M. Malakhov, CR Bot 1.15, 2000 [Online]. Available: http://arton.cunst.net/quake/crbot/
[27] R. R. Feltrin, Eraser Bot 1.01, 2000 [Online]. Available: http://downloads.gamezone.com/demos/d9862.htm
[28] ICE Bot 1.0, jibe, 1998 [Online]. Available: http://ice.planetquake.gamespy.com/
[29] K.-T. Chen, A. Liao, H.-K. K. Pao, and H.-H. Chu, "Game bot detection based on avatar trajectory," in *Proc. Int. Conf. Entertain. Comput.*, 2008, pp. 94–105.
[30] V. N. Vapnik, *The Nature of Statistical Learning Theory*, 2nd ed. New York: Springer-Verlag, 1999.
[31] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining Knowl. Disc.*, vol. 2, no. 2, pp. 121–167, 1998.
[32] T. F. Cox and M. A. A. Cox, *Multidimensional Scaling*, 2nd ed. London, U.K.: Chapman & Hall/CRC, 2000.
[33] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *J. Edu. Psychol.*, vol. 24, pp. 417–441, 1933.
[34] Y.-J. Lee and O. L. Mangasarian, "SSVM: A smooth support vector machine for classification," *Comput. Optim. Appl.*, vol. 20, no. 1, pp. 5–22, 2001.
[35] G. Shakhnarovich, T. Darrell, and P. Indyk, *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. Cambridge, MA: MIT Press, 2006.
[36] B. Schölkopf and A. Smola, *Learning With Kernels Support Vector Machines, Regularization, Optimization and Beyond*. Cambridge, MA: MIT Press, 2002.
[37] K.-T. Chen, H.-K. K. Pao, and H.-C. Chang, "Game bot identification based on manifold learning," in *Proc. ACM NetGames*, 2008, pp. 21–26.

Hsing-Kuo Pao received the B.S. degree in mathematics from National Taiwan University, Taipei, Taiwan and the M.S. and Ph.D. degrees in computer science from New York University, New York.

From 2001 to 2003, he was a Postdoctorate Research Fellow at the University of Delaware, Newark, and later he joined Vita Genomics as a Research Scientist. In 2003, he joined the Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan, as an Assistant Professor. His current research interests are machine learning, and its various applications including user behavior analysis, trajectory analysis, intrusion detection, pattern recognition, and bioinformatics.



Kuan-Ta Chen (S'04–M'06) received the B.S. and M.S. degrees in computer science from National Tsing-Hua University, Hsinchu, Taiwan, in 1998 and 2000, respectively, and the Ph.D. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2006.

Currently, he is an Assistant Research Fellow at the Institute of Information Science and the Research Center for Information Technology Innovation (joint appointment) of Academia Sinica, Taipei, Taiwan. His research interests include Internet measurement, quality-of-experience (QoE) management, network security, and online games. Much of his recent work focuses on human factors in network systems, including QoE measurement, user perception and behavior modeling, and QoE-aware system design.

Dr. Chen is a member of the Association for Computing Machinery (ACM) and IICM.

Hong-Chung Chang received the M.S. degree in computer science and information engineering from National Taiwan University of Science and Technology, Taipei, Taiwan, in 2008.

Currently, he is a Researcher at the Institute for Information Industry (III), Taipei, Taiwan. His interests include game bot detection, user behavior in online games, trajectory analysis, online games, etc.