Bot Detection in Rhythm Games: A Physiological Approach

Ruei-Min Lin[1], Hwai-Chung Ho[2], and Kuan-Ta Chen[1*]

[1]Institute of Information Science, Academia Sinica
[2]Institute of Statistical Science, Academia Sinica

ABSTRACT

As the online game industry expands, detecting and preventing cheating in games is an increasingly important research topic. Some forms of cheating, such as the use of game bots (auto-playing game clients), are particularly challenging to identify because game bots do not violate any of the game rules; rather, they simply mimic human behavior to play the game without human intervention. The use of bots introduces fairness issues to online games, and therefore robust schemes for detecting game bots are strongly demanded.

In this paper, we tackle with bots in rhythm games, which feature gameplay that incorporates eye and body coordination with music, usually a popular song. Bot detection in rhythm games is especially challenging compared with in other game genres because little information is available to distinguish the responses made by a human player from a bot. Based on the long-memoryness of the time series formed by human players' response errors to stimuli, we propose a scheme to detect the presence of human coordination mechanisms during gameplay. Based on a set of traces collected from human players and real-life game bots, we show that our scheme can accurately detect the use of game bots despite of game difficulty levels.

Categories and Subject Descriptors

H.1.2 [**Models and Principles**]: User/Machine Systems—*Human factors*; K.8.0 [**Personal Computing**]: General—*Games*

General Terms

Human Factors, Management, Security

Keywords

Game bots, Game cheating, Dancing games, Long-memory process, Long-range dependence, Human coordination

*Corresponding Author. Contact: ktchen@iis.sincia.edu.tw

Short presentation, ACE'2011 - Lisbon, Portugal

1. INTRODUCTION

In recent years, online gaming has become an important Internet activity. However, the growth of online games is accompanied by serious cheating problems [12], where one of the greatest threats that online games face today is the widespread use of game bots [2]. A game bot is a general name for describing a mechanism that can help players perform routine tasks and enhance players' performance. For instance, in MMORPGs (Massively Multi-player Online Role Player Games), players must kill monsters repeatedly to earn virtual reward and items. Bots can perform such tasks automatically without the attendance of humans. As a result, a player can save much time by using bots, while honest players must undergo significant efforts repeating such tasks manually. Similarly, in FPS (First Person Shooting) games, honest players need to manually aim and shoot their enemies with guns, while bot users can use bots to target enemies automatically and win games easily [3,9]. Likewise, in rhythm games, players have to synchronize their responses (e.g., to hit a specific key) with music, while bots can be used to automatically respond according to the rhythm and achieve a perfect timing. Such uses of game bots cause unfairness in the game world—bot users tend to be more resourceful and powerful than honest players, which makes the latter unhappy and even becomes one of the major reasons of players' quit from a game. Given all these reasons, detecting the use of game bots is a critical and demanding task of online game operators.

Generally, there are two common approaches to prevent and/or detect the use of bots: HIPs (Human Interactive Proofs) and HOPs (Human Observational Proofs). **HIPs** ask players some questions that can be answered easily by humans but difficult for bots. For example, Golle et al. [6] suggested to detect bots by using CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart), a challenge-response test which ensures that the response is given by a person. However, HIPs have two disadvantages when applied to bot detection. First, HIPs interrupt a player's game experience by querying questions. Second, HIPs can only prove that a human is present, but cannot determine if a game bot is actually used. In the case of rhythm games, it is common that bot users stay in front of the computers to watch the bot's gameplay. As a result, HIPs are not effective for detecting bot use in rhythm games.

On the other hand, **HOPs** observe and analyze players' behavior statistically to decide if the behavior is performed by human or bots. For example, Thawonmas et al. [11] identified MMORPG bots by analyzing the type and frequency

Figure 1: A screen shot of Guitar Hero

of player actions. Since HOPs do not interrupt gameplay and are capable of detecting bots no matter humans are present or not, we consider HOPs more competent than HIPs to detect rhythm game bots. However, since a player's behavior may vary, the great challenge would be how to build a robust player behavior model independent of time and various confounding factors such as music types.

In this paper, we proposes a HOP method to detect bots in rhythm games based on the long-range dependence in human coordination [4]. Specifically, when humans are asked to respond to periodic stimuli, *it is shown that the time series formed by the error between the stimulus and human response would involuntarily form a long-memory process.* We show that humans tend to exhibit the aforementioned behavior when playing rhythm games, as the rule in such games is very similar—it requires the players to respond to semi-periodic stimuli. Based on this finding, we propose a scheme that relies on the long-range dependence characteristics in human coordination behavior to detect whether a game bot is in use. Through a series of experiments, we show that our scheme can accurately discriminate game bots from human players. To the best of our knowledge, this work is the first to detect the use of game bots for rhythm games.

The remainder of this paper is organized as follows. We review related works in Section 2 and provide a brief introduction of rhythm games and long memory processes in Section 3. We conduct a pilot study, as described in Section 4, to validate the long memory characteristics of human response errors in rhythm games and consequently present our scheme in Section 5. Finally, we present the performance evaluation of our methodology in Section 6 and draw our conclusions in Section 7.

2. RELATED WORK

Game developers and operators have been eager to fight against game bots to maintain the fairness among players in a game. In this section, we review previous attempts to tackle this problem.

A common approach for preventing bots is to use CAPTCHA tests [6]. However, as we discuss in Section 1, this approach is not appropriate for rhythm games because administering such tests would interrupt gameplay and the tests can ensure only the presence rather than gameplay of humans. The HOP approach for game bot detection mostly relies on statistical behavioral modeling of human players and/or game bots. For example, Chen et al. [2] proposed to identify standalone MMORPG bots by analyzing network traffic patterns, whereas the traffic generated game bots tend to contain a certain degree of regularity. A number of researches proposed to detect game bots based on

Figure 2: A screen shot of Gitaroo Man

players' behavior from different aspects, such as item trading behavior [11], how the input devices (e.g., mouse and keyboard) are controlled [5], the avatars' moving trajectory over time [8, 9], and so on.

Unfortunately, all the previous proposals cannot be applied to rhythm games because the rule and control in this type of games are very simple and thus no much information can be utilized to develop sophisticated statistical models for distinguishing bots and human players. Specifically, compared to other game genres, there is only one action (e.g., dancing) for game avatars and only one type of control (e.g., hitting an arrow key) required from players. Thus, the only information available for discriminating bots and human players are the correctness and the timing of the arrow key events.

3. BACKGROUND

In this section, we first introduce rhythm games and their relationship with game bots; we then give a recap of long memory processes and how the long-memoryness of a time series is determined.

3.1 Rhythm Games

Rhythm games feature gameplay that incorporates eye and body coordination with music, usually a popular song. To score well, a player must translate visual and auditory cues into actions and perform them at appropriate time and in rhythm. Themed experiences through custom hardware controllers, such as dancing on a game pad or playing a guitar shaped controller, are popular as well.

Early on, icons streaming across the screen in a timeline fashion were used to indicate the time of a joystick tilt or a button press, including Rock Band, Guitar Hero (Figure 1), and Dance Dance Revolution. Rather than on a timeline, another style of rhythm games arranges the icons radially around the screen, where icons emerge from the middle of the screen and project outward with different angles. Gameplay of rhythm games sometimes involves an analog thumb stick. Notable examples in this category are Gitaroo Man (Figure 2) and EyeToy: Groove. Moreover, a new form of rhythm gameplay evolved incorporating the absolute position of the Nintendo DS touch screen. In the Japanese game Osu! Tatakae! Ouendan (Figure 3), players must tap circles, which move along a trace line, in the correct order with correct timings according to the rhythm. In brief, the essence of rhythm gameplay is that the players strive to syn-

Figure 3: A screen shot of Osu! Tatakae! Ouendan

chronize their responses to stimuli at the right timing and that the smaller the timing and positioning errors between the responses and stimuli, the higher score a player would achieve.

Bots in Rhythm Games

In rhythm games, players usually compete with each other by achieving higher scores/ranks or collecting more valuable virtual items. Achieving such goals require players to continuously deliver outstanding performance in gameplay, which is challenging and requires numerous practice and efforts. Therefore, dishonest players may use bots to play games for them, where the bots automatically respond according to the stimuli and the rhythm. The use of game bots certainly annoys honest players because the latter may be more skillful and spend much more time, but rank lower than bot users. Furthermore, developing bots for rhythm games is relatively easier and less troublesome than developing bots for games of other genres, as stimuli and controls are particularly simple. Therefore, the use of bots are now prevalent in rhythm games and the detection of such unlawful behavior is strongly demanded.

However, detecting the use of bots in rhythm games is particularly difficult due to two reasons. First, rhythm game players tend to be present when they use bots to play the game, as they can interact with other players during gameplay. Second, unlike other game genres, the stimuli from the game are low-dimensional as they comprise only direction and time information. In other words, there is little information available to discriminate the responses made by a human player, especially an experienced player, from a game bot.

3.2 Long Memory Process

Our proposed scheme for bot detection is based on an observation that the time differences between a human's responses to a periodic stimulus tend to form a long memory process [4]. In this section, we provide a short recap of the long memory process, including the definition of long memory processes and the estimation of the Hurst index, which can be used to determine whether a time series is a long memory process or not.

Below we recap the mathematical definition of long memory processes [1]. Let X_t be a stationary process for which there exists a real number $\alpha \in (0,1)$ and a constant $c_\rho > 0$ such that

$$\lim_{k\to\infty} \rho(k)/[c_\rho k^{-(1-\alpha)}] = 1, \quad (1)$$

where $\rho(k)$ denotes the auto-correlations of X_t at lag k, then X_t is called a stationary process with long memory or long range dependence.

To explain plainly, a long memory process is a process with a random component, where a past event has a decaying effect on future events. The process has some memory of past events, which is gradually forgotten as time moves forward.

Hurst Index

The Hurst index [7], commonly denoted as H, is an important indictor of the long memory property of a time series. H measures the relative tendency of a time series either to regress strongly to the mean or to cluster towards a direction. A value of $0 < H < 0.5$ indicates a time series with a negative autocorrelation (i.e., a decrease between values will probably be followed by an increase). A value of $0.5 < H < 1$ indicates a time series with a "long-term" positive autocorrelation (i.e., an increase between values will probably be followed by another increase). A value of $H = 0.5$ indicates a process of white noise, where it is equally likely that a decrease or an increase will follow any particular value. That is, a time series has no memory of previous values if its Hurst index is 0.5.

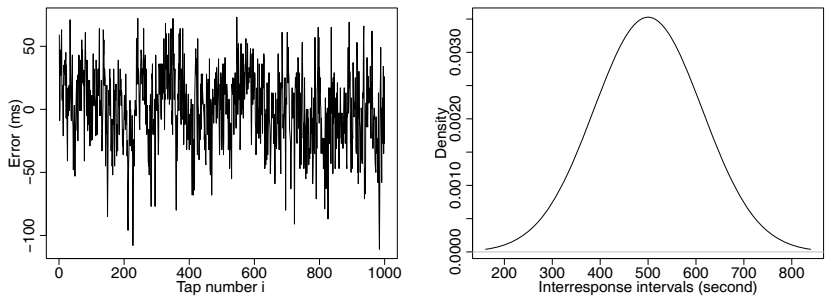
There are quite a few methods available for estimating the Hurst index of a time series [1]. We use the following two common approaches for Hurst index estimation:

1. **Spectrum:** Given a time series e_i and its spectral density function $S(f)$. If e_i is a long memory process, the spectral density function will follow a power law, i.e., $S(f) \sim f^{-\alpha}$, where α is in the range of $(0,1)$. We can therefore estimate H based on α using the equation $H = (1+\alpha)/2$.
2. **R/S plot:** Given a time series e_i, $L(n,s) = \Sigma_{i=1}^{i=s} e_{n+i}$ can be regarded as the position of a random walk after s steps, and $R(n,s) = max\{L(n,p) - p \cdot L(n,s)/s, 1 \le p \le s\} - min\{L(n,p) - p \cdot L(n,s)/s, 1 \le p \le s\}$ can be regarded as the trend-corrected range of the random walk. Moreover, $S^2(n,s)$ denotes the sample variance of the data set $\{e_{n+i}\}_{i=1}^{i=s}$. Assuming $Q(s)$ the average re-scaled statistic $Q(s) = \langle R(n,s)/S(n,s)\rangle_n$, we can plot s versus $Q(s)$ in a logarithmic scale and estimate the Hurst index as the slope of the linear regression model $log(Q(s)) \sim log(s)$.

As a demonstration, assuming that we are to estimate the Hurst index of the time series in Figure 4(a). According to the spectrum method, we plot the spectral density function of the time series in Figure 4(b), and estimate its Hurst index as $(1+0.69)/2 = 0.84$. By using the R/S plot approach, we estimate the Hurst index of the time series as 0.72, as shown in Figure 4(d).

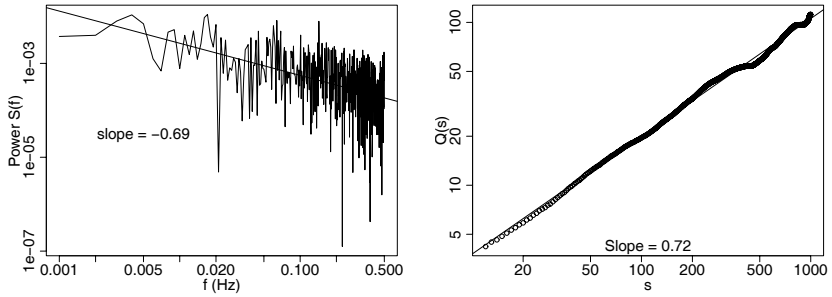
4. PILOT STUDY

In the experiments conducted by Chen et al. [4], subjects are asked to synchronize their limb movements to periodic stimuli. Because of the internal mechanisms of human coordination, the time series formed by the errors between the subjects' responses (i.e., movements) and the stimuli



(a) The response error time series

(b) Density function of inter-response intervals



(c) Spectral density function of the response error series

(d) R/S plot of the response error series

Figure 4: An exemplar trace from the monotonic stimulus pilot study

are shown to be long memory processes by estimating their Hurst indexes. For brevity, we shall call the time series formed by the timing errors between a subject's responses to stimuli the "response error time series" or simply the "response error series."

In this section, we conduct two pilot studies in order to validate whether the long-range dependence characteristics in human coordination can be generalized to rhythm games. Generally, the gameplay of rhythm games is similar to the experiment settings in [4]; however, a few differences exist: 1) The stimuli in rhythm games are not fully periodic, rather they are multiples of a base frequency. For example, if the base frequency is 16 Hz (equivalent to a period of 62.5 ms), the periods between two stimulus can be 62.5 ms, 125 ms, 187.5 ms, 250 ms, and so on. 2) Rhythm games are decorated by dazzling images and background music, which may affect how players coordinate their key press actions in response to stimuli. To confirm whether the response error series still exhibits the long memory property, we conduct two studies, the first with monotonic (i.e., periodic) stimuli, and the second with harmonic (i.e., the frequencies of stimuli are multiples of a base frequency) stimuli, and investigate whether the long-memory property still holds in these conditions.

4.1 Monotonic Stimulus Experiment

Our monotonic stimulus experiment is exactly the reproduction the experiment conducted in [4]. In this study, the subjects are prompted to click the SPACE key whenever they hear a crisp sound signal that is triggered every 500 ms. Each experiment lasts 10 minutes and comprises $1,200$ signals and ideally $1,200$ responses if the subjects respond correctly. We record the occurrence times of each stimulus (sound signal) and a subject's response (key press) and consequently compute the response error time series by the differences between stimuli and responses.

We plot the results from one of the experiments in Figure 4. The response error time series is shown in Figure 4(a), while it can be seen from Figure 4(b) that the stimuli are triggered every 500 ms as the inter-response times concentrate around 500 ms. Interestingly, both graphs indicate that the average response error is slightly below zero, which implies that humans tend to "respond" to periodic stimuli based on their internal coordination mechanisms rather than completely relying on the external stimuli. This phenomenon can be considered an evidence of human coordination mechanisms in work. We estimate the Hurst index of the response error series by using the two methods introduced in Section 3.2. The Hurst indexes estimated are $(1 + 0.69)/2 = 0.84$ (Figure 4(c)) and 0.72 (Figure 4(d)) respectively. In either case, the Hurst index is in the range of $0.5 < H < 1.0$, which indicates that the response error series is a long memory process and reconfirms the observation made by [4].

4.2 Harmonic Stimulus Experiment

Having confirmed the long memory property using monotonic stimulus experiments, we relax the periodicity requirement of the stimuli in order to match the gameplay of rhythm games and conduct harmonic stimulus experiment. In this study, the stimuli are not triggered in a fully periodic manner; instead, they are triggered with a random inter-stimuli period, which can be 250 ms, 500 ms, or 1000 ms. In addition, we mimic rhythm games by providing visual cues in addition to auditory cues. The visual cues are presented as icons moving along a horizontal band; whenever an icon reaches a designated area, the sound signal is simultaneously triggered to prompt the subjects to respond.

We plot the results from one of the harmonic stimulus experiments in Figure 5. The response error time series is shown in Figure 5(a), and the density function of the inter-response intervals are depicted in Figure 5(b), which manifests that the three periods, 250 ms, 500 ms, and 1000 ms, occur with approximately equal probabilities. We estimate the Hurst index of the response error series by using the spectrum approach as $(1 + 0.34)/2 = 0.67$, as shown in Figure 5(c); also, we estimate the Hurst index as 0.82 based on the R/S plot, as shown in Figure 5(d). Both Hurst index estimations indicate that the response error series is a long memory process. Therefore, we can confirm that, even with harmonic stimuli and visual cues, the long memory property of response error series still holds. This phenomenon will serve the basis of our scheme for detecting bots in rhythm games.

5. PROPOSED SCHEME

In this section, we propose our scheme for detecting bots in rhythm games. To detect the bots, we collect the information about stimuli and a player's corresponding responses, and determine whether the "player" is a human or a bot based on whether the response error series is a long memory process. In the following, we first discuss the derivation of the response error series and then present the classifier for bot identification.

In a rhythm game, players are prompted with harmonic stimuli (using visual and auditory cues) which are normally associated with different types of responses. For example, a common design is to have four types of stimuli that correspond to the four orthogonal directions, up, right, down, and left, respectively, while players are expected to hit the corresponding arrow key when a stimulus is present. The four-direction setting can be augmented by including four diagonal directions and forms an eight-direction setting to increase the variety and also the difficulty of the games.

Assuming that the i-th stimulus of type $c(s_i)$ was triggered at time $t(s_i)$, the j-th response of type $c(p_j)$ was made by a player at time $t(p_j)$, the response error associated with the j-th response, p_j, is computed by $e_j = t(p_j) - t(s_i)$ that satisfies

$$i = \underset{k \in [1,n] \wedge c(p_k) = c(s_k)}{argmin} |t(p_k) - t(s_k)|, \quad (2)$$

where n is the number of stimuli we observed in a game session. By so doing, we can collect the response error series $\{e_i\}_{i=1}^{i=N}$.

Based on the observation in Section 3.2, for a recorded game session, our game bot detection method comprises three steps:

1. Compute the response error time series $\{e_i\}$.
2. Estimate the Hurst index by using both the spectrum method and the R/S plot method.
3. If both of the estimated Hurst indexes indicate that $\{e_i\}$ possesses the long memory property, we consider that the game session was played by a human; otherwise, it was played by a game bot.

6. PERFORMANCE EVALUATION

In this section, we evaluate whether the Hurst index of a response error series is feasible to be a robust indicator of human gameplay. We begin by introducing the studied game, *Dancing Online*, and then present how we collect the data from the game. We then perform a preliminary analysis of the collected traces, and finally investigate the overall discrimination performance of the proposed scheme.

Figure 6: A screen shot of *Dancing Online*

6.1 Studied Game

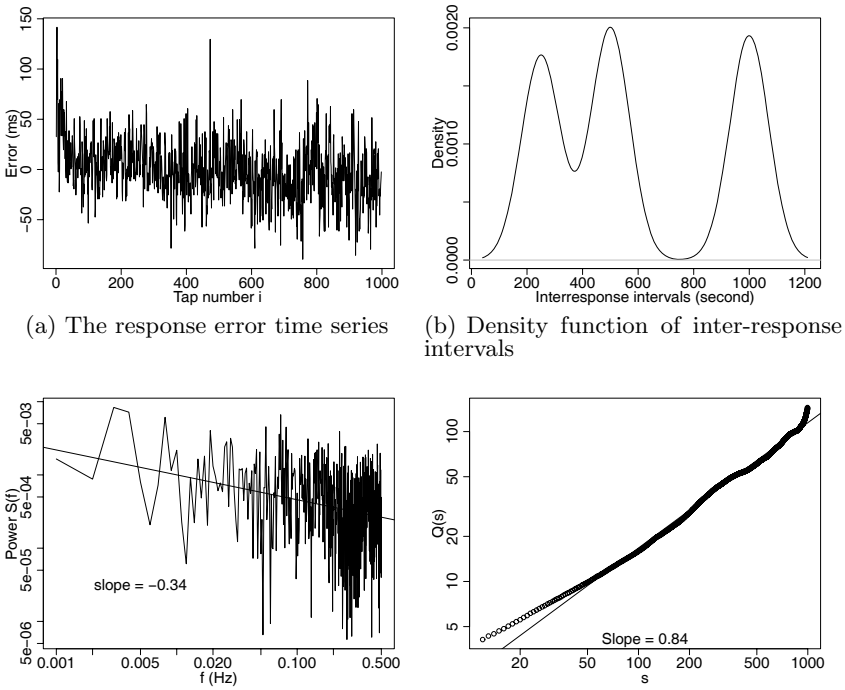
Our performance evaluation was based on a popular rhythm game in Asia, *Dancing Online*[1]. *Dancing Online* provides a variety of gameplay modes, including the normal mode, beat mode, rotation mode, and so on. We chose the beat mode for data collection because it is the most popular mode in the game. In this mode, a stimulus which can be in one of the four directions moves along a band from left to right, as shown in Figure 6. When a stimulus reaches the target area (the lightened area at the right side of the screen), the player needs to hit the corresponding arrow key to score points. The smaller the timing difference between the time the stimulus reaches the target area and the time the arrow key is hit, the higher score is obtained.

We chose *Dancing Online* for study due to three reasons. First, it is a popular game that there are normally thousands of players online at any time. The large player population makes it a good candidate for testing our methodology in real life. Second, *Dancing Online* is a typical rhythm game with a number of variations in gameplay, which allows us to evaluate the effect of human coordination when more confounding factors are put into play. Third, game bots designed for *Dancing Online* are easily accessible; therefore, we can validate the effectiveness of our methodology using real-life game bots, rather than using home-grown bots that may not be realistic.

6.2 Bots for Dancing Online

So far (as of Sep 2011), the most popular bot for *Dancing Online* is called `Dancing King`. `Dancing King` is a standalone program which monitors the game screen and emulates key presses whenever appropriate. Because players whose performance is "too good" tend to be suspected by other players and game operators, `Dancing King` supports an anti-detection mechanism which purposely makes error by tuning a value called ADV (Anti-Detection Value). With this mechanism enabled, `Dancing King` does not always respond perfectly; instead, they may make timing errors in a probabilistic way. Because of the anti-detection mechanism,

[1] `http://www.gamespot.com/pc/puzzle/dance/`



(a) The response error time series

(b) Density function of inter-response intervals

(c) Spectral density function of the response error series

(d) R/S plot of the response error series

Figure 5: An exemplar trace from the harmonic stimulus pilot study

it becomes harder for a game operator to confirm whether a player is using `Dancing King` to cheat or not.

6.3 Data Collection

Dancing Online is a close-sourced, proprietary, commercial game which does not support any form of built-in trace collection facility. To record the information of stimuli and players' responses, we utilize the hooking mechanism[2] in Windows to inject our instrumentation code into the *Dancing Online* client. We use the `detours` library to intercept the `IDirect3dDevice9::EndScene()` function, which is called when a Direct3D application finishes drawing graphics on a hidden surface and is about to present the surface on the screen. By analyzing the screen content, we continuously record the stimulus type, $c(s_i)$, on the screen and the time it arrives at the target area, $t(s_i)$, for the i-th stimulus. In addition, we intercept the window procedure of the main window of *Dancing Online*, which is called whenever a window message is sent to the main window and about to be processed. By examining the window messages, we record the type and timing of each arrow key press event made by the player as $c(p_i)$ and $t(p_i)$ for the i-th response. Because our injection approach is general, it can be applied to other rhythm games easily without much modification.

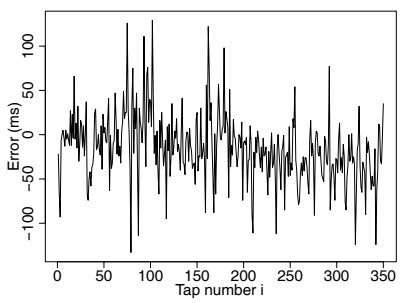
We hired 11 gamers with different levels of experience in *Dancing Online* to play the game with our data logging facility enabled. Also, we used the bot program `Dancing King` to play *Dancing Online* and recorded its gameplay behavior. In a total, we have collected 466 samples from 11 human players and 172 samples from the bot. Each sample comprises the title of the song, the game difficulty level, and four time series, namely, $\{t(s_i)\}$, $\{c(s_i)\}$, $\{t(p_i)\}$, and $\{c(p_i)\}$, which are used to calculate $\{e_i\}$ according to the procedure in Section 5. The game difficulty can be one of the four levels, Easy, Medium, Advanced, and Hardcore.

[2]The Windows hooking mechanism is invoked by calling the `SetWindowsHookEx` function. It is frequently used to inject code into other processes.

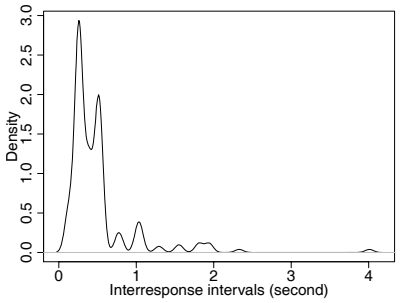
6.4 Data Inspection

In this subsection, we analyze the traces collected during gameplay to inspect the behavioral differences between human players and bots. Firstly, we plot the response error time series of a human gameplay sample and its associated graphs in Figure 7. The response error series shown in Figure 7(a) validates our observation in Section 4 that players tend to respond earlier than the time prompted. Figure 7(b) depicts the density function of the inter-response intervals, which indicates that the stimuli are not periodic but concentrate around several values. Using the spectrum method, we estimate the Hurst index as $(1 + 0.4)/2 = 0.7$, as shown in Figure 7(c); in addition, the Hurst index estimated using the R/S plot method is 0.84, as shown in Figure 7(d). Both estimators strongly indicate the long memory property of the response error series and support our argument about the human coordination behavior in rhythm games.

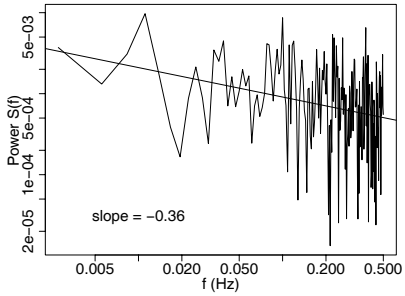
As a comparison, we also plot the response error time series of a bot gameplay sample and its associated graphs in Figure 8. From Figure 8(a), we can see that the response errors generated by bots are more like white noises rather than human coordination errors that exhibit certain patterns, such as the premature response behavior. The Hurst indexes estimated are $(1 - 0.16)/2 = 0.42$ and 0.46 according to the spectrum method (Figure 8(c)) and the R/S plot method (Figure 8(d)) respectively. Both estimators indicate that the response error series generated by bots is not a long memory process, because a bot does not have complex coordination mechanisms like humans which cause long-range



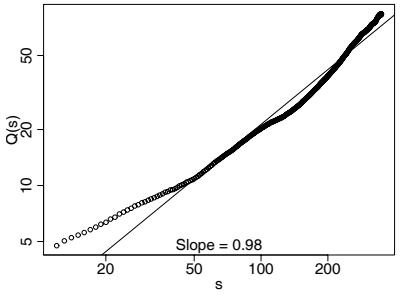
(a) The response error time series



(b) Density function of inter-response intervals

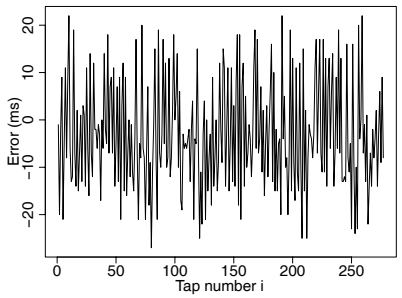


(c) Spectral density function of the response error series

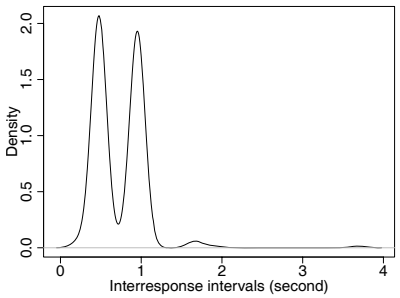


(d) R/S plot of the response error series

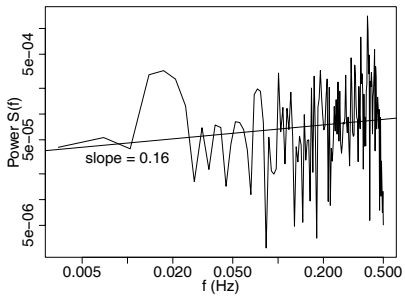
Figure 7: An exemplar human player trace collected from *Dancing Online*



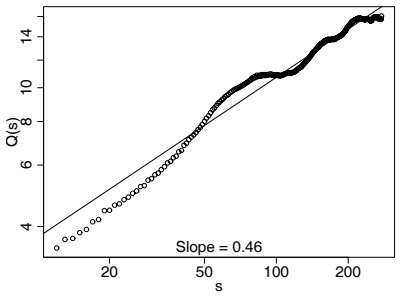
(a) The response error time series



(b) Density function of inter-response intervals



(c) Spectral density function of the response error series



(d) R/S plot of the response error series

Figure 8: An exemplar bot trace from *Dancing Online*

Table 1: Average Hurst indexes of response error series generated by human players and bots respectively. The third row (AUC) stands for the Area Under Curve, which quantifies the power when the Hurst index is used to discriminate human players from bots under each game difficulty level.

	Easy	Medium	Adv.	Hardcore	Overall
Human	0.588	0.613	0.613	0.567	0.601
Bot	0.507	0.454	0.459	0.472	0.463
AUC	0.805	0.952	0.970	0.843	0.924

dependent timing errors.

6.5 Classification Performance

Among our traces, the average Hurst index of human samples is 0.601 and that of bot samples is 0.463. We shall use AUC (Area Under Curve) to quantify whether the Hurst index can serve a good predictor for discriminating human player and bot samples. A brief recap of the ROC (Receiver Operating Characteristic) curve and AUC (Area Under Curve) [10] is given as follows. In the signal detection theory, a ROC curve is a graphical plot of TPR (True Positive Rate) versus FPR (False Positive Rate) for a binary classifier over different discrimination thresholds. AUC is the area under an ROC curve, which can be used to indicate the discrimination power of a binary classifier:

- AUC ≈ 0.5: No discrimination;
- $0.7 \leq$ AUC < 0.8: Acceptable discrimination;
- $0.8 \leq$ AUC < 0.9: Good discrimination;
- AUC ≥ 0.9: Outstanding discrimination.

We list the average Hurst indexes of the response error series generated by human players and bots, as well as the AUC for classifying the categories in Table 6.5. In addition, the ROC curves with different game difficulty levels are plotted in Figure 9. From the table, the overall AUC over all difficulty levels is above 92%, which indicates that the Hurst index alone is a powerful discriminating factor to classify human players and bots. We find that bots' behavior remain roughly the same regardless of the game difficulty levels. In contrast, the long memory property of human response errors tend to be less prominent when the game difficulty is the most simple or the most difficult. It is reasonable because when the game is very simple, human players can perfectly follow the stimuli without much effort in coordination. On the other hand, if the game is so difficult that the coordination mechanism cannot keep up with the stimuli, humans tend to make more mistakes under such condition and their responses would contain more randomness.

7. CONCLUSION

In this paper, we have proposed a novel scheme to detect bots in rhythm games. We have shown that when humans synchronize their responses to harmonic stimuli, the series formed by the time differences between the responses and stimuli tend to form a long memory process. Based on this property, the proposed scheme examines the long-memoryness of the response error series to determine whether a game session is played by humans or bots. Using a set of traces collected from a number of players and real-life bot programs, we have shown that the scheme can detect the use of game bots despite of game difficulty levels.

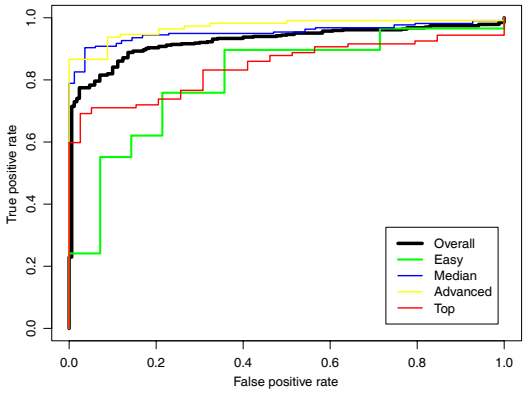


Figure 9: The ROC curves of our Hurst-index-based classifier with different game difficulty levels

Acknowledgments

This work would not have been possible without the support from International Games System Co. Ltd., the developer company of *Dancing Online*. Moreover, the authors are much indebted to Andrew Liao, who developed the trace collection program for *Dancing Online*. This work was supported in part by the National Science Council under the grant NSC100-2628-E-001-002-MY3.

8. REFERENCES

[1] J. Beran. *Statistics for long-memory processes*. Monographs on Statistics and Applied Probability. 61. London: Chapman Hall. x, 315 p. 45.00 , 1994.

[2] 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 Journal on Advances in Signal Processing*, 2009.

[3] K.-T. Chen, H.-K. K. Pao, and H.-C. Chang. Game bot identification based on manifold learning. In *Proceedings of ACM NetGames 2008*, 2008.

[4] Y. Chen, M. Ding, and J. A. S. Kelso. Long memory processes ($1/f^{\alpha}$ type) in human coordination. *Phys. Rev. Lett.*, 79(22):4501–4504, Dec 1997.

[5] S. Gianvecchio, Z. Wu, M. Xie, and H. Wang. Battle of botcraft: fighting bots in online games with human observational proofs. In *Proceedings of ACM CCS'09*, CCS '09, pages 256–268. ACM, 2009.

[6] P. Golle and N. Ducheneaut. Preventing bots from playing online games. *Comput. Entertain.*, 3:3–3, July 2005.

[7] B. B. Mandelbrot and J. W. Van Ness. Fractional brownian motions, fractional noises and applications. *SIAM Rev.*, 10:422–437, 1968.

[8] S. Mitterhofer, C. Kruegel, E. Kirda, and C. Platzer. Server-side bot detection in massively multiplayer online games. *IEEE Security and Privacy*, 7:29–36, 2009.

[9] H.-K. Pao, K.-T. Chen, and H.-C. Chang. Game bot detection via avatar trajectory analysis. *IEEE Transactions on Computational Intelligence and AI in Games*, Sep 2010.

[10] J. A. Swets. Signal detection theory and ROC analysis in psychology and diagnostics: Collected papers. 1996.

[11] R. Thawonmas, Y. Kashifuji, and K.-T. Chen. Detection of MMORPG bots based on behavior analysis. In *Proceedings of ACM ACE 2008*, 2008.

[12] J. Yan and B. Randell. A systematic classification of cheating in online games. In *Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, NetGames '05, pages 1–9, New York, NY, USA, 2005. ACM.