Game Bot Detection Based on Avatar Trajectory*

Kuan-Ta Chen[1], Andrew Liao[2], Hsing-Kuo Kenneth Pao[3], and Hao-Hua Chu[2]

[1] Institute of Information Science, Academia Sinica
[2] Dept. of Computer Science & Information Engineering, National Taiwan University
[3] Dept. of Computer Science & Information Engineering, National Taiwan Univ. of Science & Technology

Abstract. In recent years, online gaming has become one of the most popular Internet activities, but cheating activity, such as the use of game bots, has increased as a consequence. Generally, the gaming community disagrees with the use of game bots, as bot users obtain unreasonable rewards without corresponding efforts. However, bots are hard to detect because they are designed to simulate human game playing behavior and they follow game rules exactly. Existing detection approaches either interrupt the players' gaming experience, or they assume game bots are run as standalone clients or assigned a specific goal, such as aim bots in FPS games.

In this paper, we propose a trajectory-based approach to detect game bots. It is a general technique that can be applied to any game in which the avatar's movement is controlled directly by the players. Through real-life data traces, we show that the trajectories of human players and those of game bots are very different. In addition, although game bots may endeavor to simulate players' decisions, certain human behavior patterns are difficult to mimic because they are AI-hard. Taking Quake 2 as a case study, we evaluate our scheme's performance based on real-life traces. The results show that the scheme can achieve a detection accuracy of 95% or higher given a trace of 200 seconds or longer.

Key words: Cheating Detection, Online Games, Quake, Security, Supervised Classification, User Behavior

1 Introduction

In recent years, online gaming has become 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 artificial intelligence that players use for different

* This work was supported in part by Taiwan Information Security Center (TWISC), National Science Council under the grants NSC 97-2219-E-001-001 and NSC 97-2219-E-011-006. It was also supported in part by Taiwan E-Learning & Digital Archives Program (TELDAP), National Science Council under the grants NSC 96-3113-H-001-010 and NSC 96-3113-H-001-012.

purposes. In MMORPGs (Massively Multiplayer Online Role Player Games), 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. In FPS (First-Person Shooter) games, users can employ bots to play in place of themselves in order to get high scores and gain a reputation in the community.

Generally, the gaming community disagrees with 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 [1, 2] propose using CAPTCHA tests during a game to determine whether an avatar is actually controlled by a person. Although this method is effective, it interrupts the game play and degrades players' feelings of immersion in the virtual world [3, 4]. Alternatively, passive detection approaches, such as schemes based on traffic analysis [5, 6] and schemes based on avatars' shooting accuracy in FPS games [7], can be used. The former approach assumes that a game bot works as a standalone client, and the latter is only valid for detecting aim bots in shooting games.

In this paper, we propose a general approach for *all genres of games where players control the avatar's movement directly.* Our approach is based on the *avatar's movement trajectory* during a game. The rationale is that *the trajectory of the 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 information provided in the game. Since human decisions may not always be logical and efficient, how to model and simulate realistic movements is still an open question in the AI field. To distinguish human players from game bots efficiently, we analyze the trajectories of both player types and distinguish between the trajectories according to their *spatial and temporal characteristics.* We choose 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 two-fold. 1) We propose a trajectory-based approach for detecting game bots. It is a general model that can be applied to any game in which the avatar's movement is controlled by the players directly. 2) Using real-life human traces, the performance evaluation results show that the scheme can achieve a detection accuracy of 95% or higher when the trace length is 200 seconds or longer. Because it is difficult to simulate human players' logic when they control game characters, we believe 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 2 contains a review of related works. In Section 3, we introduce our game case study, Quake 2, and describe the game trace collection methodology. We analyze the similarities and differences between the trajectories of different types of players in Section 4. In Section 5, we propose an identification scheme and demonstrate its ability in

terms of the distribution of discriminative features. In Section 6, we evaluate the performance of the proposed scheme with the consideration of the trace length. Then, in Section 7, we summarize our conclusions.

2 Related Work

Recently, anti-cheating software programs, such as PunkBuster and GameGuard, have been widely deployed in online games to prevent cheating. Such software is bundled with game clients, so it cannot be uninstalled even if the game client has been removed. It works by hiding in the game client process, monitoring the entire virtual memory space (to prevent modification of the game's executable images), blocking suspected programs that might be hacker tools, and blocking certain API calls. This kind of software can detect nearly all plug-in tools that attach to a game client program to inspect or modify game states when the game is running. Unfortunately, it cannot stop the widespread use of standalone bots, including the bot series we study in this paper. The reason is that these anti-cheating software programs are host-based, so they must be installed on players' PCs to be effective. Standalone bots, on the other hand, can function *without* clients, and it is unlikely that anti-cheating tools would be installed on PCs where the bots are running. This claim is strongly supported by the fact that game bots are still active in games protected by PunkBuster or GameGuard, e.g., Quake (PunkBuster) and Lineage[4] (GameGuard).

3 Data Description

3.1 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 via 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 community.

To ensure that our game traces represented the diversity of Quake players, we only used traces that players had contributed voluntarily. The human players' traces were downloaded from the following archive sites: GotFrag Quake[5], Planet Quake[6], Demo Squad[7], and Revilla Quake Site[8]. We restricted the traces to the map *The Edge*, one of the most well-known levels of death-match play. On this map, the only goal is that each player should kill as many other players

[4] `http://boards.lineage2.com/showflat.php?Number=573737`
[5] `http://www.gotfrag.com/quake/home/`
[6] `http://planetquake.gamespy.com/`
[7] `http://q2scene.net/ds/`
[8] `http://www.revilla.nildram.co.uk/demos-full.htm`

Table 1. Trace Summary

	name	num	mean	total	active
1	Human	93	2 hour	203.5 hour	91%
2	CR	24	19 hour	448.8 hour	91%
3	Eraser	15	20 hour	296.4 hour	94%
4	ICE	18	20 hour	358.8 hour	67%

as possible, until the time limit is reached. Because short traces contain little information, we only collected traces longer than 600 seconds.

3.2 Bot Traces

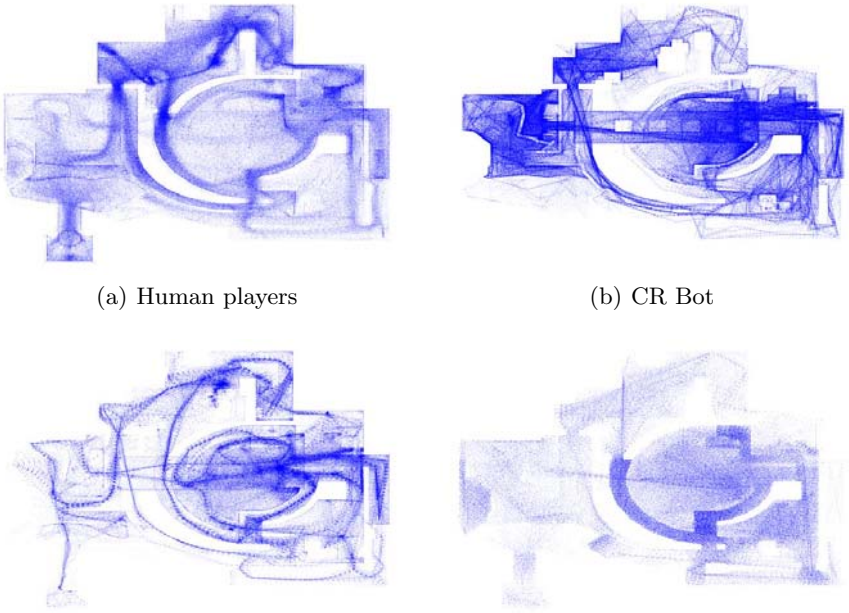
There are many game bots available for Quake 2. For this study, we selected three of the most popular bot programs for trace collection, namely CR BOT 1.14 [8], Eraser Bot 1.01 [9], ICE Bot 1.0 [10]. We collected $1,306$ hours of traces in total, as shown in Table 1. In CR Bot and Eraser Bot, all human players and bots were active most of time ($\geq 90\%$). There was less activity in ICE Bot because it often remained idle in some places waiting for an opportunity to ambush other players.

4 Discriminative Analysis

In this section, we compare the avatar trajectories of human players and game bots. First, we compare the navigation patterns of the two player types and consider their individual trajectories. We then identify the most significant discriminative characteristics of the respective trajectories and incorporate them into the proposed bot identification scheme.

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 graph, as shown in Fig. 1. The areas of high density in each figure are the places that players visit more frequently, while the sparse areas represent buildings or other types of obstacles that players cannot pass. The figures show that the game level is formed by squares, plazas, and narrow corridors. This arrangement is designed specifically for death-match play, as the winding routes provide cover for players to hide, and the narrow corridors lead to intense fighting if players confront each other in these places. We observe that, even though all the movement traces were collected on the same map, the navigation patterns of different player types are dissimilar. We summarize the differences below.

1. Human players tended to explore all areas on the map; thus, Fig. 1(a) shows the most complete terrain of the level. In contrast, the routing algorithms of game bots may have had difficulty navigating to some places, so they never visited some parts of the map. For example, the bottom left-hand corner of



(a) Human players (b) CR Bot

(c) Eraser Bot (d) ICE Bot

Fig. 1. Presence locations of all players

the CR Bot navigation map in Fig. 1(b) does not indicate the presence of bots.

2. To reduce the probability of being attacked, human players normally avoid open spaces. Therefore, in Fig. 1(a) we observe that human players avoided the plaza in the middle of the map, and stayed in the surrounding corridors instead. This is indicated by the high density of plots in the corridors. In contrast, game bots often stay in the central plaza, probably because it occupies a large space and it is easy to get everywhere from this area.
3. Even though human players spend most of their time in narrow areas and confined rooms, there are large variations in their trajectories. There are two reasons for this phenomenon. 1) The width of the main routes is quite large. Rather than stay in the middle of a route, players move irregularly within the limited space. 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. 2) As fights may occur anytime-anywhere, human players often move strategically to dodge current or potential attacks. On the other hand, we find that different game bots adopt very different movement patterns over the routes. The movement paths of CR Bot and Eraser Bot (Fig. 1(b) and Fig. 1(c) respectively) are dense and easy to see. This suggests that these bots tend to follow exact movement patterns when moving through the same corridor. In contrast, ICE

Bot (Fig. 1(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 roughly equivalent.

Clearly, there are substantial differences between the aggregated navigation patterns of human players and those of each game bot because the bots' routing patterns are very different from the movement behavior exhibited by human players.

5 Bot Detection Scheme

Our objective is to classify human players and game bots efficiently and accurately. To this end, we integrate the spatial and temporal differences in the trajectories of avatars controlled by different player types to develop a bot identification scheme. In this section, we first describe the set of discriminative features extracted from the avatar trajectories, and then explain how we use the features to classify game bots and human players.

5.1 Feature Extraction

Given a segment of a trajectory, $\{x_t, y_t\}, 1 \leq t \leq T$, we extract the following features from this two-dimensional time series.

1. ON/OFF Activity First, we note that avatars in the game play do not move all the time. Sometimes they may stop to check if any opponents are around, wait for opponents to enter an area, wait for regeneration of their weapons or ammunition, or simply take a rest. The alternate moving and idle behavior forms an ON/OFF movement pattern. We define ON periods as consecutive periods of movement longer than 1 second, and OFF periods as the remaining time frames. The duration and frequency of ON/OFF periods are decided by the players' styles and the bots' AI logic. For example, aggressive players may keep moving all the time, while cautious players may stay in one place to monitor their surroundings. Therefore, we define four features based on ON/OFF activity: *the mean and standard deviation of ON periods, and those of OFF periods.*

Fig. 2 shows the distributions of the four features. The mean and standard deviation of human players' ON periods are significantly higher than those of game bots. This indicates that human players are more aggressive as they tend to move all the time. In addition, the mean and standard deviation of human players' OFF periods are longer than those of bots, which implies that human behavior is more irregular and unpredictable in that they may wait for a longer time after a long move. The figure shows that human players and game bots differ in terms of ON/OFF activity. Hence, we believe that the four features based on these activities could be useful for bot detection.

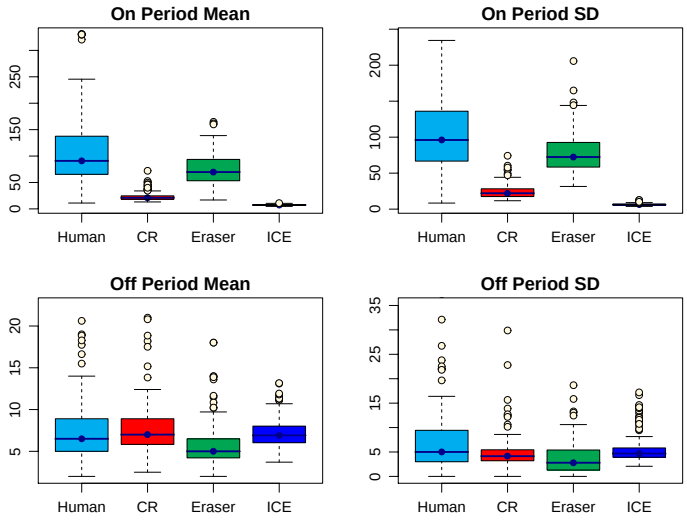


Fig. 2. The distribution of features related to ON/OFF periods.

2. Pace In games, avatars are generally allowed to move at different speeds and in different ways, such as running, slow walking, step-by-step walking, lateral shifting, and moving backwards. In addition, players can stop the current movement and proceed with another movement in different direction in sub-seconds; therefore, the resulting avatar movements can be highly variable. One simple way to characterize the dynamics of an avatar's movement is by the pace of its movements. We define the *pace* as the displacement of an avatar's coordinate in one second, and extract *the mean and standard deviation of the pace* as two features. We find that the paces of most avatars are generally small, although they can be large occasionally. To characterize the variability of paces when players move fast, we also define *the "large pace SD," which is the standard deviation of paces larger than* 10 *units*.

In addition to normal movements, players may teleport their avatars to a remote place instantly through a teleportation spot. Teleportation may also be used when an avatar dies. It will be transferred to the rebirth spot so that its life points can be recharged. We detect teleportation occurrences by computing if the offset in one second is longer than 60 units and define the feature "teleportation rate" as *the average count of teleportation occurrences recorded in one second*.

Fig. 3 shows the distribution of the four features related to the movement pace and teleportation. Although the means of the paces of different player types are dissimilar, the variations are not large. This shows that the four player types have different but consistent micro-movement behavior in small time scales. The standard deviation of the pace also has large discriminability, where that of human players and Eraser Bot have similar magnitude. The large standard

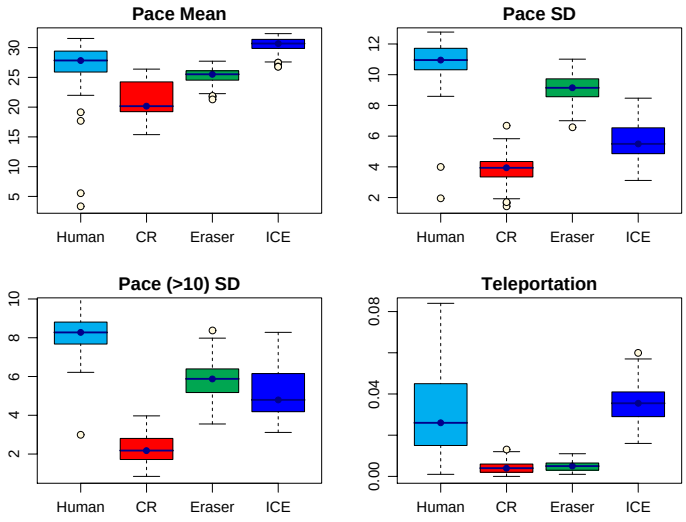


Fig. 3. The distribution of features related to movement pace.

deviation of the pace, on the other hand, exhibits great discriminability, which indicates that human players have even larger pace variability when they move fast. Finally, CR Bot and Eraser Bot have very low teleportation frequency. In contrast, human players have moderate teleportation frequency. Moreover, their variance is high because human players have different preferences when using teleportation spots and players get killed at different rates.

3. Path We also define the following features to characterize the detailed trajectories of avatars in a game.

Lingering. We consider whether players "lingered" in a small area during a specific time period. For an avatar at (x, y) at time t, if its distance from (x, y) was always less than d during the period $(t, t + p)$, we say that the avatar was lingering during $(t, t + p)$, given the parameters (d, p). We arbitrarily set $d = 30$ seconds and $p = 300$ units, as we find that different parameters do not affect the classification performance significantly.

Smoothness. The "smoothness" feature determines whether an avatar moves in straight or zig-zag patterns. Assume an avatar is at (x_1, y_1) at time t_1 and at (x_2, y_2) at time t_2. We define the smoothness as the number of times the character moves across the line $(x_1, y_1) - (x_2, y_2)$ during the period (t_1, t_2). As the line $(x_1, y_1) - (x_2, y_2)$ indicates the shortest route between the two points (x_1, y_1) and (x_2, y_2), crossing the line implies that the player is moving inefficiently. This may be because he is attempting to dodge gunfire, switch to another target, or simply due to players' habits or bots' routing algorithms.

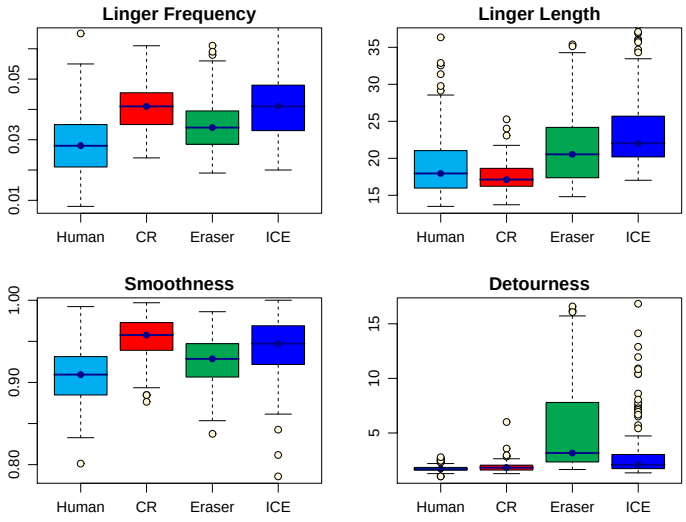


Fig. 4. The distribution of features related to movement path.

Detour. We define another feature "detour" to quantify the effectiveness of user movements. If an avatar is at (x_1, y_1) at time t_1 and at (x_2, y_2) at time t_2, we compute the detour by dividing the length of the movement by the effective offset of an avatar during the period (t_1, t_2).

The distributions of the above features are plotted in Fig. 4. The graph shows that the linger frequency and duration of human players are significantly less than those of game bots. This is reasonable because lingering in a place for a long time is a dangerous, as the player may be noticed and induce opponents' fire. The smoothness of human players is the lowest of the four player types, which supports the intuition that human players' movements are the most irregular and unpredictable. The detour feature shows that Eraser Bot moves very inefficiently in terms of the avatar's effective offset. In contrast, the movements of human players are relatively more efficient. We suspect this is because *human players tend to move away from current positions to another place efficiently even though they may move irregularly and strategically; thus, the resulting avatar trajectory exhibits both unpredictability and efficiency which seem contradictory.*

4. Turn Our final set of features is based on the frequency and amplitude of how avatars change direction. Our rationale is that each time an avatar changes direction, the magnitude of the change should be dependent on player conventions and bot routing algorithms.

Assume an avatar is at (x_1, y_1) at time t, at (x_2, y_2) at time $t + p$, and at (x_3, y_3) at time $t + 2p$. If the angle between two vectors $(x_2 - x_1, y_2 - y_1)$ and $(x_3 - x_1, y_3 - y_1)$ is greater than a, we determine that a turn with angle a

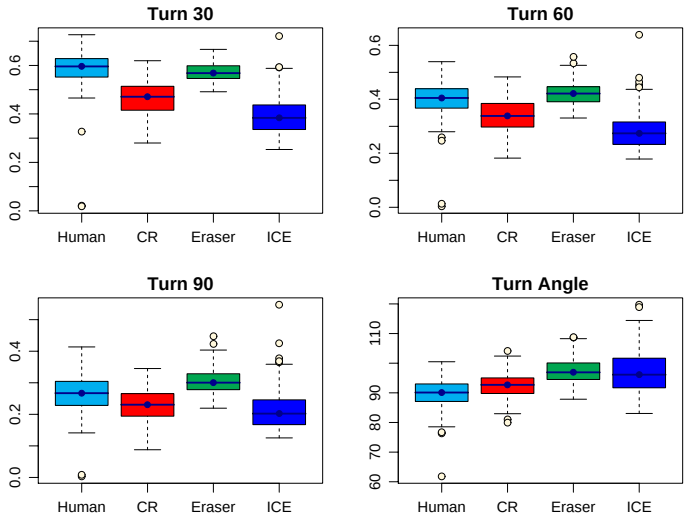


Fig. 5. The distribution of features related to turn movement.

occurred. We define three features to denote the frequency of turns with angles 30°, 60°, and 90°, respectively. In addition, we define a feature called the "turn angle" to denote the average angle change for all direction changes greater than 30°.

Fig. 5 shows the distributions of the turn-related features. We observe that the four player types change direction at different rates no matter how we define the minimum degree of a direction change. Notably, the turn frequency of human players is the highest for the 30° angle and becomes relatively lower for the 90° angle. In addition, the average turn angle of human players is the lowest among the four types, which indicates that *human players tend to adjust their directions continuously and slightly.*

5.2 Classification

We apply a supervised classification framework to train a classifier, which we use to determine whether a segment of an avatar's trajectory belongs to a human player or a game bot. The classifier we adopt is the naive Bayesian classifier without the kernel density estimation. We evaluate the performance of trajectory classification in the next section.

6 Performance Evaluation

In this section, we evaluate the performance of our proposed bot detection scheme on the collected traces. First, we evaluate whether our scheme can distin-

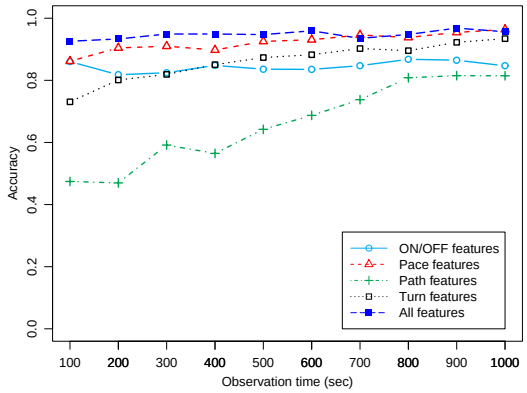


Fig. 6. Classification accuracy between human and bots.

guish between human players and game bots, by using the classifier to perform 10-fold cross-validation. In real-life scenarios, the trace length plays an important role because it determines how quickly a game bot can be detected. Thus, we evaluated the performance of our scheme on different traces lengths, as shown in Fig. 6. The graph shows that the detection accuracy is higher than 90%, even when the trace length is as short as 100 seconds. Longer traces yield better accuracy. To determine which category of features yields the highest accuracy, we plot the classification performance for each category of features. The results indicate that the features related to the movement pace, direction changes, and ON/OFF periods all yield good results, while path-related features only exhibit good discriminability when the trace length is 800 seconds or longer.

Furthermore, we perform a player-type classification; that is, we not only determine whether a character is controlled by a human player or a bot program, but also identify the bot program used if appropriate. The results are shown in Fig. 7. The classification accuracy of the player types is even better than that of the human-bot scenario when the trace length is longer than 200 seconds. With a trace length of 500 seconds or longer, our scheme yields a classification accuracy of 98% or higher. However, in this setting, individual feature categories, except those related to movement paces, exhibit low discriminability when they are applied to the classification separately.

7 Conclusion

We have proposed a trajectory-based approach for detecting game bots. It is a general technique that can be applied to any game in which the avatar's movement is controlled by the players directly. Our analysis of real-life traces shows that the trajectories of human players and game bots are very dissimilar. The performance evaluation results show that our bot detection scheme can achieve

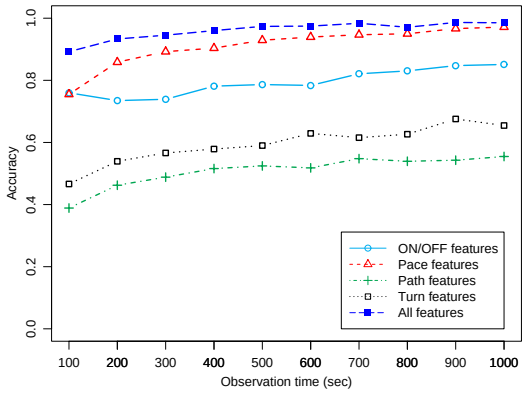


Fig. 7. Classification accuracy between four types of players (human and three bot programs).

a detection accuracy of 95% or higher when the trace length is 200 seconds or longer. Because it is difficult to simulate human players' behavior when controlling game characters, we believe our method has the potential to distinguish between human players and automated programs, and thus merits further investigation.

References

1. Golle, P., Ducheneaut, N.: Preventing bots from playing online games. Computers in Entertainment **3**(3) (2005) 3–3
2. von Ahn, L., Blum, M., Hopper, N.J., Langford, J.: CAPTCHA: Using hard AI problems for security. In: Proceedings of Eurocrypt. (2003) 294–311
3. Novak, T.P., Hoffman, D.L., Duhachek, A.: The influence of goal-directed and experiential activities on online flow experiences. Journal of Consumer Psychology **13**(1) (2003) 3–16
4. Ila, S., Mizerski, D., Lam, D.: Comparing the effect of habit in the online game play of australian and indonesian gamers. In: Proceedings of the Australia and New Zealand Marketing Association Conference. (2003)
5. Chen, K.T., Jiang, J.W., Huang, P., Chu, H.H., Lei, C.L., Chen, W.C.: Identifying MMORPG bots: A traffic analysis approach. In: Proceedings of ACM SIGCHI ACE'06, Los Angeles, USA (Jun 2006)
6. Chen, K.T., Huang, P., Lei, C.L.: Game traffic analysis: An MMORPG perspective. Computer Networks **50**(16) (2006) 3002–3023
7. Yeung, S., Lui, J., Liu, J., Yan, J.: Detecting cheaters for multiplayer games: theory, design and implementation. Proc IEEE CCNC **6** 1178–1182
8. Malakhov, M.: CR Bot 1.15 (May 2000) http://arton.cunst.net/quake/crbot/.
9. Feltrin, R.R.: Eraser Bot 1.01 (May 2000) http://downloads.gamezone.com/demos/d9862.htm.
10. jibe: ICE Bot 1.0 (1998) http://ice.planetquake.gamespy.com/.