Game Bot Identification based on Manifold Learning

Kuan-Ta Chen  Academia Sinica
Hsing-Kuo Pao  NTUST
Hong-Chung Chang  NTUST
Game Bots

- Game bots: automated AI programs that can perform certain tasks in place of gamers

- Popular in MMORPG and FPS games
  - MMORPGs (Role Playing Games)
    - accumulate rewards in 24 hours a day
    - break the balance of power and economies in game
  - FPS games (First-Person Shooting Games)
    - a) improve aiming accuracy only
    - b) fully automated
    - achieve high ranking without proficient skills and efforts
Bot Detection

- Detecting whether a character is controlled by a bot is difficult since a bot obeys the game rules perfectly.
- No general detection methods are available today.

- State of practice is identifying via human intelligence:
  - Detect by “bots may show regular patterns or peculiar behavior”
  - Confirm by “bots cannot talk like humans”
  - Labor-intensive and may annoy innocent players.
Related Work

- **Prevention**
  - CAPTCHA (reverse Turing tests) [Golle et al; 2005]

- **Detection**
  - Process monitoring at client side [GameGuard]
    - Bot program’s signatures are keeping changing
  - Traffic analysis at the network [Chen et al; 2006]
    - Remove bot traffic’s regularity by heavy-tailed random delays
  - Aiming bot detection using DBN [Yeung et al; 2006]
    - Specific to aiming bots that help aim the target accurately
CAPTCHA in a Japanese Online Game
Our Goal of Bot Detection Solutions

- **Passive** detection
  - No intrusion in players’ gaming experience

- **No client software support** is required

- **Generalizable** schemes (for other games and other game genres)
Our Solution: Trajectory + Manifold Learning

- Based on the avatar’s movement trajectory in game
- Applicable for all genres of games where players control the avatar’s movement directly
- Avatar’s trajectory is high-dimensional (both in time and spatial domain)
  - Use manifold learning to distinguish the trajectories of human players and game bots
The Rationale behind Our Scheme

- The trajectory of the avatar controlled by a human player is hard to simulate for two reasons:
  - **Complex context information:**
    Players control the movement of avatars based on their knowledge, experience, intuition, and a great deal of environmental information in game.
  - **Human behavior is not always logical and optimal**
- How to model and simulate realistic movements (for game agents) is still an open question in the AI field.
Bot Detection: A Decision Problem

Q: Whether a bot is controlling a game client given the movement trajectory of the avatar?
A: Yes / No?
Talk Progress

- Overview
- Data Description
- Proposed Scheme
  - Pace vector construction
  - Dimension Reduction using Isomap
  - Classification
- Performance Evaluation
- Conclusion
Choose Quake 2 as our case study

- A classic FPS game
- Many real-life human traces are available on the Internet
  ➔ more realistic than traces collected in experiments
A Screen Shot of Quake 2
Data Collection

- Human traces downloaded from fan sites including GotFrag Quake, Planet Quake, Demo Squad, and Revilla Quake Site
- Bot traces collected on our own Quake server
  - CR BOT 1.14
  - Eraser Bot 1.01
  - ICE Bot 1.0
- Totally 143.8 hours of traces were collected

<table>
<thead>
<tr>
<th>Name</th>
<th>Number</th>
<th>Trace Length</th>
<th>Total</th>
<th>Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>282</td>
<td>1000 seconds</td>
<td>78.0 hours</td>
<td>89%</td>
</tr>
<tr>
<td>CR</td>
<td>75</td>
<td>1000 seconds</td>
<td>20.8 hours</td>
<td>89%</td>
</tr>
<tr>
<td>Eraser</td>
<td>102</td>
<td>1000 seconds</td>
<td>28.3 hours</td>
<td>92%</td>
</tr>
<tr>
<td>ICE</td>
<td>60</td>
<td>1000 seconds</td>
<td>16.7 hours</td>
<td>67%</td>
</tr>
</tbody>
</table>
Aggregate View of Trails (Human & 3 Bots)

Human

CR Bot

Eraser

ICE Bot
Trails of Human Players
Trails of Eraser Bot
Trails of ICE Bot
Talk Progress

- Overview
- Data Description
- Proposed Scheme
  - Pace vector construction
  - Dimension Reduction using Isomap
  - Classification
- Performance Evaluation
- Conclusion
The Complete Process: Overview

1. **Pace Vector Construction**
   - Step 1. Pace Vector Construction

2. **Dimension Reduction with Isomap**
   - Step 2. Dimension Reduction with Isomap

3. **Supervised classification**
   - Step 3. Supervised classification

- Send CAPTCHA To Users
- Decision
Step 1. Pace Vector Construction

- For each trace $s_n$, we compute the pace (distance) in successive two seconds by

$$
\|s_{n,i+1} - s_{n,i}\| = \sqrt{(s_{n,i+1} - s_{n,i})^T(s_{n,i+1} - s_{n,i})}
$$

- We then compute the distribution (histogram) of paces with a fixed bin size by

$$
F_n = (f_{n,1}, f_{n,2}, \ldots, f_{n,B})
$$

where $B$ is the number of bins in the distribution.
Pace Vector: An Example

(A) Human

(B) Bot

B is set to 200 (dimensions) in this work
Step 2. Dimension Reduction with Isomap

- We adopt Isomap for nonlinear dimension reduction for
  - Better classification accuracy
  - Lower computation overhead in classification

- Isomap
  - Assume data points lie on a manifold
    
    A mathematical space in which every point has a neighborhood which resembles Euclidean space, but in which the global structure may be more complicated. (Wikipedia)

  - 1. Construct the neighborhood graph by kNN (k-nearest neighbor)
  - 2. Compute the shortest geodesic path for each pair of points
  - 3. Reconstruct data by MDS (multidimensional scaling)
A Graphic Representation of Isomap

(A) A Swiss Roll Data  
(B) Neighborhood Graph  
(C) After Mapping by Isomap
PCA (Linear) vs. Isomap (Nonlinear)

(A) PCA

(B) Isomap
Step 3. Classification

- Apply a supervised classifier on the Isomap-reduced pace vectors
  - SVM (Support Vector Machine) in our study
- To decide whether a trajectory belongs to a game bot or a human player
Talk Progress

- Overview
- Data Description
- Proposed Scheme
  - Pace vector construction
  - Dimension Reduction using Isomap
  - Classification
- Performance Evaluation
- Conclusion
## Five Methods for Comparison

<table>
<thead>
<tr>
<th>Method</th>
<th>Data Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>kNN</td>
<td>Original 200-dimension Pace Vectors</td>
</tr>
<tr>
<td>Linear SVM</td>
<td></td>
</tr>
<tr>
<td>Nonlinear SVM</td>
<td></td>
</tr>
<tr>
<td>Isomap + kNN</td>
<td>Isomap-reduced Pace Vectors</td>
</tr>
<tr>
<td>Isomap + Nonlinear SVM</td>
<td></td>
</tr>
</tbody>
</table>
Evaluation Results

Error Rate

False Positive Rate

False Negative Rate
Addition of Gaussian Noise

- Bot programmers can try to evade from detection by adding random noise into bots’ movement behavior

- Evaluate the robustness of our schemem by adding Gaussian noise into bots’ trajectories
Evaluation Results

![Graph showing error rate, false positive rate, and false negative rate across different noise levels for various algorithms.](image)

- **Error Rate**
- **False Positive Rate**
- **False Negative Rate**

Kuan-Ta Chen / Game Bot Identification based on Manifold Learning
Cross-Map Validation

- Human movement may be restricted by the environment around him/her
- Whether a classifier trained for a map can be used for detecting bots on another map?
Evaluation Results

Error Rate

False Positive Rate

False Negative Rate
Talk Progress

- Overview
- Data Description
- Proposed Scheme
  - Pace vector construction
  - Dimension Reduction using Isomap
  - Classification
- Performance Evaluation
- Conclusion

Kuan-Ta Chen / Game Bot Identification based on Manifold Learning
Conclusion

- We propose a trajectory-based approach for detecting game bots.

- The results show that the Isomap + nonlinear SVM approach performs good and stable results.

- Human’s logic in controlling avatars is hard to simulate ➔ we believe this approach has the potential to be a general yet robust bot detection methodology
Future Work

- Include more **spatial-domain information** in the pace vector
- Validate our methodology on other games (game genres)
Thank You!

Kuan-Ta Chen

http://www.iis.sinica.edu.tw/~ktchen