Game Bot Detection
Based on Avatar Trajectories

Kuan-Ta Chen
Academia Sinica

Andrew Liao
NTU

Hsing-Kuo Kenneth Pao
NTUST

Hao-Hua Chu
NTU
Game Bots

- AI programs that can perform some tasks in place of gamers

- Popular in MMORPG and FPS games
  - MMORPGs (Massive Multiplayer Role Playing Games)
    accumulating rewards efficiently in 24 hours a day
    ➔ breaking the balance of power and economies in games
  - FPS games (First-Person Shooting Games)
    a) improving aiming accuracy
    b) fully automated
    ➔ achieving 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:
  - **Detection:** Bots may show regular or peculiar behavior;
    **Confirmation:** Bots cannot talk like humans.
  - Labor-intensive and may annoy innocent players.
Earlier Bot Prevention / Detection Work

- **Prevention**
  - CAPTCHA tests
    - (Completely Automated Public Turing test to tell Computer and Human Apart)

- **Detection**
  - Process monitoring at client side
    - Constantly changing the bot program’s signature
  - Traffic analysis at the network
    - Remove bot traffic’s regularity by heavy-tailed random delays
  - Aiming bot detection using Dynamic Bayesian Network
    - Specific to aiming bots that help aim the target accurately
CAPTCHA in a Japanese Online Game
Our Goal for Bot Detection

- Passive detection
  → No intrusion in players’ gaming experience
- No client software is required
- Generalizable schemes (for other games and other game genres)
Our Solution: Trajectory-based Detection

- Based on the avatar’s moving 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 domains)
  - 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 information provided in the game.
  - **Human is not always logical and optimal**

- How to model and simulate realistic movements 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 Schemes
  - Feature-based
  - Manifold learning:
    Dimensionality Reduction using Isomap

- Conclusion
Case Study: Quake 2

- 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 GotFrag Quake, Planet Quake, Demo Squad, and Revilla Quake Sites
- Bot traces collected on our own Quake server
  - CR BOT 1.14
  - Eraser Bot 1.01
  - ICE Bot 1.0
- Each cut into 1,000-second segments
- 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>
## Data Representation

<table>
<thead>
<tr>
<th>Time (sec.)</th>
<th>Trace</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$S_1$</td>
</tr>
<tr>
<td>1</td>
<td>164.87, -258.87</td>
</tr>
<tr>
<td>2</td>
<td>159.87, -259.87</td>
</tr>
<tr>
<td>3</td>
<td>157.66, -264.42</td>
</tr>
<tr>
<td>i</td>
<td>527.87, 788.00</td>
</tr>
<tr>
<td>t</td>
<td>984.00, 192.00</td>
</tr>
</tbody>
</table>

$t \ (X, Y) \ (X, Y) \ (X, Y)$
Trails of Human Players
Trails of CRBot
Trails of Eraser Bot
Trails of ICE Bot
Talk Progress

- Overview
- Data Description
- Proposed Schemes
  - Feature-based
  - Manifold learning:
    - Dimensionality Reduction using Isomap
- Conclusion
Proposed Schemes

- Scheme 1 – **Feature Based**: Extraction of Important Features, Combined with **Decision Tree**
  - ON/OFF Activity
  - Pace
  - Path
  - Turn

- Scheme 2 – **Manifold Learning Based**
  - High-dimensional data
  - Dimension reduction
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:
  - ON/OFF Activity
  - Pace
  - Path
  - Turn
Movement Trail Analysis

- **Activity**
  - mean/sd of ON/OFF periods

- **Pace (1 step)**
  - speed/offset in each time period
  - teleportation frequency

- **Path (several steps)**
  - linger frequency/length
  - smoothness
  - detourness

- **Turn**
  - frequency of mild turns, wild turns, and U-turns
ON/OFF Activity Features
Pace-Related Features

**Pace Mean**

- **Human**
- **CR**
- **Eraser**
- **ICE**

**Pace SD**

- **Human**
- **CR**
- **Eraser**
- **ICE**

**Pace (>10) SD**

- **Human**
- **CR**
- **Eraser**
- **ICE**

**Teleportation**

- **Human**
- **CR**
- **Eraser**
- **ICE**
Path-Related Features

\[ \text{detourness} = \frac{\text{path length}}{\text{offset}} \]
Turn-Related Features

Turn 30

Turn 60

Turn 90

Turn Angle

Hsing-Kuo Kenneth Pao / Game Bot Detection Based on Avatar Trajectory
Data Point Mapping using the First Three Principle Components
Classification Results using Decision Tree

![Graph showing accuracy over observation time for different trajectory features and all features combined.](image)
Proposed Schemes

- **Scheme 1 – Feature Based:** Extraction of Important Features, Combined with Decision Tree
  - ON/OFF Activity
  - Pace
  - Path
  - Turn

- **Scheme 2 – Manifold Learning Based**
  - High-dimensional data
  - Dimension reduction
The Complete Process: Overview

User Trajectory → Preprocess → Transform to Pace Vector & Discretization

→ Dimension Reduction

→ Isomap

→ Classification

→ k NN & SSVM

→ Decision
Step 1. Pace Vector Construction

- For trace $s_n$, we compute the pace in consecutive 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 of pace lengths 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.
Examples of Pace Vectors

(A) Human

(B) Bot
33 Hsing-Kuo
Kenneth Pao / Game Bot Detection Based on Avatar Trajectory

Step 2. Dimension Reduction with Isomap

- Pace vectors have 201 dimensions
- We adopt Isomap for dimension reduction for
  - Better performance
  - Reduce computation overhead in classification
- Isomap
  - Assume data points lie on a smooth manifold
  - Construct the neighborhood graph by $k$NN ($k$-nearest neighbors)
  - Compute the shortest path for each pair of points
  - Reconstruct data by MDS (multidimensional scaling)
A Graphic Representation of Isomap

(A) A Swiss Roll Data  (B) Neighborhood Graph

(C) After Mapping by Isomap
Distance Measure between Pace Vectors

1. Euclidean Distance

2. Kullback-Leibler divergence (or KL distance)

\[ d_{KL}(P(x)||Q(x)) = \sum_{i=1}^{d} P(x_i) \log \frac{P(x_i)}{Q(x_i)} \]

where \( P \) & \( Q \) are feature vectors of dimension \( d \)

- Because KL distance is not symmetric, we use a symmetric version instead

\[ D_{KL}(P(x)||Q(x)) = \frac{d_{KL}(P(x)||Q(x)) + d_{KL}(Q(x)||P(x))}{2} \]
PCA (Linear) vs. Isomap (Nonlinear)

(A) PCA

(B) Isomap
Step 3. Classification

- Apply a supervised classifier on the Isomap-reduced pace vectors
- To decide whether a pace vector belongs to a bot or a human player
# Five Methods for Comparison

<table>
<thead>
<tr>
<th>Method</th>
<th>Data Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k$NN</td>
<td>Pace Vectors</td>
</tr>
<tr>
<td>Linear SVM</td>
<td>Pace Vectors</td>
</tr>
<tr>
<td>Nonlinear SVM</td>
<td>Isomap-reduced Pace Vectors</td>
</tr>
<tr>
<td>Isomap + $k$NN</td>
<td>Isomap-reduced Pace Vectors</td>
</tr>
<tr>
<td>Isomap + Nonlinear SVM</td>
<td>Isomap-reduced Pace Vectors</td>
</tr>
</tbody>
</table>
Evaluation Results

Error Rate

False Positive Rate

False Negative Rate
Talk Progress

- Overview
- Data Description
- Proposed Schemes
  - Feature-based
  - Manifold learning:
    Dimensionality Reduction using Isomap
- Conclusion
Conclusion

- We propose a trajectory-based approach for detecting game bots.
- Feature-based method shows descent result.
- We can improve the result and efficiency further by Isomap + nonlinear SVM approach.
- 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.
Questions?
Thank You!

Hsing-Kuo Kenneth Pao
Addition of Gaussian Noise

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

- Evaluate the robustness of our scheme by adding Gaussian noise into bots’ trajectories.
Addition of Gaussian Noise (cont)

Human

Gaussian Noise

Bots

Bots with Noise
Evaluation Results

Error Rate

False Positive Rate

False Negative Rate
Cross-Maps Validation

- Sometimes 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