Trajectory based Behavior Analysis for User Verification

Hsing-Kuo Pao Ph.D.
National Taiwan University of Science and Technology

A joint work with
Hong-Yi Lin, Kuan-Ta Chen, and Junaidillah Fadlii
Motivation

- Many of our activities on computer need a verification step for authorized access.
  - E.g.: bank transactions,
  - talking with friends,
  - playing on-line games, and so on.
- We would like to propose an approach for user verification, (and recognition if possible), …
Some Typical Approaches

- Password/Pin number
- Hardware: ATM cards, magnetic strip, …
- Biometrics: fingerprint, iris, face, voice, …
- CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) for bot detection
Common CAPTCHA Schemes for Bot Prevention

CAPTCHA in a Japanese Online Game

Hsing-Kuo Pao / The 11th International Conference on Intelligent Data Engineering and Automated Learning, 2010
Some Typical Approaches

- Password/Pin number
- Hardware: ATM cards, magnetic strip, …
- Biometrics: fingerprint, iris, face, voice, …
- CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart)
- How about user’s input trajectories?
Trajectory Input

Online game: human user

Mouse trace (left-handed)

Mouse trace (right-handed)

Online game: bot

Hsing-Kuo Pao / The 11th International Conference on Intelligent Data Engineering and Automated Learning, 2010
Based on User Trajectories...

- We propose a scheme to verify whether a user is the true account owner based on the user’s trajectories

Verification based on trajectories:

- **Passive** verification
  - No bothering to users’ activities
- **Hard to simulate/duplicate**: an AI-hard problem?
- May serve as a **confirmation** (binary) instead of a **recognition** (multi-class) process, in a **2-factor** scheme
Notations and Inputs

- Given a trajectory \( s = (x_1, x_2, \ldots, x_t, \ldots, x_T) \) of length \( T \) with \( x_i \in \mathbb{R}^2 \) or \( \mathbb{R}^3 \), we are interested in several feature sets:

  - Trajectory step: \( v_t = x_{t+1} - x_t \)
  - Step size: \( \lambda_t = \|v_t\| \)
  - Step angle: \( \theta_t \), the angle between vector \( v_t \) and the \( x \)-axis
An Running Application: Game Bot Detection

- 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
Bot Detection: A Decision Problem

Q: Whether a bot is controlling a game client given the movement trajectory of the avatar?
A: Yes / No?
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 in Quake 2

- A classic FPS game
- Many real-life human traces are available on the Internet
  ➔ more realistic than traces collected in experiments
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>
Talk Progress

- Motivation
- Proposed problem: an running example

Our Schemes
- Entropy-based
- Without temporal information
- With temporal information

- Evaluation
- Conclusion
Trails of Human Players

Hsing-Kuo Pao / The 11th International Conference on Intelligent Data Engineering and Automated Learning, 2010
Trails of Bot (Eraser)
Preliminary Analysis: Entropy-based

- The bot’s trajectories exhibit less randomness/irregularity than the human’s trajectories.

- Measuring entropy…

  \[ H_{s_i}(X) = -P(x) \log P(x) - (1 - P(x)) \log(1 - P(x)) \]

  where \( X \) describes an event that a trace \( s \) touches a location \( i \) within a certain period.
### Entropy-based Detector

- **Result of entropy values for human and bots**

<table>
<thead>
<tr>
<th>Class</th>
<th>Human</th>
<th>Bot</th>
<th>Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entropy</td>
<td>9.33</td>
<td>8.46</td>
<td>8.56</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8.72</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>8.41</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>8.95</td>
<td></td>
</tr>
</tbody>
</table>

### Classification on 138 traces (80% training + 20% test)

<table>
<thead>
<tr>
<th>Class</th>
<th>Data</th>
<th>Human</th>
<th>Bot</th>
<th>Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numbers &amp; Results</td>
<td>Training</td>
<td>75</td>
<td>20 27 16</td>
<td>6.52%</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>19</td>
<td>5  7  4</td>
<td>11.43%</td>
</tr>
</tbody>
</table>
Talk Progress

- Motivation
- Proposed problem: an running example
- Our Schemes
  - Entropy-based
    - Without temporal information
    - With temporal information
- Evaluation
- Conclusion
Proposed Schemes

- Scheme 1 – without temporal information:
  - Considering step size distribution

- Scheme 2 – with temporal information
  - Dissimilarity measure based on Markov modeling and KL divergence
Distribution on Step Size

- We compute the discrete distribution of step size with a fixed bin size by
  \[(P_0, P_1, \ldots, P_B)\]
  where \(B\) is the number of bins in the distribution.

- KL divergence measuring “distance” of two distributions \(P\) and \(Q\)
  \[d^{SS}(P, Q) = \sum_b P(b) \log \frac{P(b)}{Q(b)}\]
Step Size of Human and Bot

(A) Human

(B) Bot

Much more randomly distributed from human traces
than from bot traces!!!
Talk Progress

- Motivation
- Proposed problem: an running example
- Our Schemes
  - Entropy-based
  - Without temporal information
  - With temporal information
- Evaluation
- Conclusion
Assume step size $\lambda_t$ and angle $\theta_t$ following Gaussian distributions

$$p(\lambda_{t+1} | \lambda_t) = \frac{1}{\sqrt{2\pi}\sigma_\lambda} \exp \left( -\frac{(\lambda_{t+1} - \lambda_t)^2}{2\sigma_\lambda^2} \right)$$

$$p(\theta_{t+1} | \theta_t) = \frac{1}{\sqrt{2\pi}\sigma_\theta} \exp \left( -\frac{(\theta_{t+1} - \theta_t)^2}{2\sigma_\theta^2} \right)$$

Given a model $\mathcal{M}$, the log-likelihood of a trace $s$ can be written as

$$\ell(s; \mathcal{M}) = \log L(s; \mathcal{M}) = \log \left( P(x_1) + \prod_{t=2}^T P(x_t | x_{t-1}) \right)$$

$$= \log P(x_1) + \sum_{t=2}^T \log P(x_t | x_{t-1})$$
Dissimilarity Measure

- Code length based on Shannon code
  \[ c(s \mid \mathcal{M}) = - \log L(s; \mathcal{M}) \]
- Given two trajectories \( s_1 \) and \( s_2 \), we define the distance between them as
  \[
  D_{MC}^{MC}(s_1, s_2) = \frac{c(s_1 \mid \mathcal{M}_2) + c(s_2 \mid \mathcal{M}_1)}{c(s_{12} \mid \mathcal{M}_{12})},
  \]
  where \( s_{12} \) is a new trace formed by concatenating the traces \( s_i \) and \( s_j \), and \( \mathcal{M}_{ij} \) is the associated model of the concatenated trace \( s_{ij} \).
Manifold Learning with 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)
PCA (Linear) vs. Isomap (Nonlinear)

The detection may be done is higher dimensional space!!!
Last Step: Classification

- Given pairwise dissimilarities, ideally, we can apply any supervised classifiers or unsupervised methods to decide whether a trajectory belongs to a bot or a human player.
- E.g., kNN, SVM, ...
Talk Progress

- Motivation

- Proposed problem: an running example

- Our Schemes
  - Entropy-based
  - Without temporal information
  - With temporal information

- Evaluation

- Conclusion
Evaluations

- Effectiveness of Proposed Methods
  - With different classification or clustering methods
  - With or without manifold learning
  - With or without temporal information
- Using Trajectories of Different Length
- With Added Noise from Camouflage
- Various Kinds of Inputs
Comparison with Different Classification or Clustering Methods

- Given the step-size inputs (i.e., without temporal information)

(Average of 10 repeats of the 10-fold cross-validation)

<table>
<thead>
<tr>
<th>Classification Methods</th>
<th>FP (%)</th>
<th>FN (%)</th>
<th>Err (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1k) kNN</td>
<td>0.00</td>
<td>3.22</td>
<td>1.45</td>
</tr>
<tr>
<td>(1sl) Linear SSVM</td>
<td>1.07</td>
<td>0.95</td>
<td>1.02</td>
</tr>
<tr>
<td>(1sn) Nonlinear SSVM</td>
<td>1.43</td>
<td>0.35</td>
<td>0.94</td>
</tr>
<tr>
<td>(2k) Isomap + kNN</td>
<td>0.00</td>
<td>1.30</td>
<td>1.16</td>
</tr>
<tr>
<td>(2sl) Isomap + Linear SSVM</td>
<td>0.00</td>
<td>25.34</td>
<td>11.43</td>
</tr>
<tr>
<td>(2sn) Isomap + Nonlinear SSVM</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Comparison with or without Temporal Information

- Given the inputs of the Markov chain-described code length (i.e., with temporal information)
- Average of 10 repeats of the 10-fold cross-validation

<table>
<thead>
<tr>
<th>Classification Methods</th>
<th>FP (%)</th>
<th>FN (%)</th>
<th>Err (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2k) Isomap + kNN</td>
<td>0.00</td>
<td>1.30</td>
<td>1.16</td>
</tr>
<tr>
<td>(2sl) Isomap + Linear SSVM</td>
<td>0.00</td>
<td>25.34</td>
<td>11.43</td>
</tr>
<tr>
<td>(2sn) Isomap + Nonlinear SSVM</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>(3k) Isomap + kNN</td>
<td>1.11</td>
<td>0.00</td>
<td>0.59</td>
</tr>
<tr>
<td>(3sl) Isomap + Linear SSVM</td>
<td>2.67</td>
<td>0.91</td>
<td>2.32</td>
</tr>
<tr>
<td>(3sn) Isomap + Nonlinear SSVM</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Without vs. With Temporal Information

(B) Step Size, Without Temporal Information

(C) Markov Chain, With Temporal Information

The detection may be done is higher dimensional space!!!
Using Trajectories of Different Length

- Step Size Inputs:

- Markov Chain Described Code:
Adding noise for camouflage

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

Without temporal information

Hsing-Kuo Pao / *The 11th International Conference on Intelligent Data Engineering and Automated Learning, 2010*
### Handwriting, Mouse Traces, Online Game and Animal Movement

- **Data Statistics on Various Data Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Number of Classes</th>
<th>Number of Instances</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handwriting</td>
<td>11</td>
<td>110</td>
<td>2D</td>
</tr>
<tr>
<td>Mouse</td>
<td>8</td>
<td>178</td>
<td>2D</td>
</tr>
<tr>
<td>Game</td>
<td>94</td>
<td>940</td>
<td>2D</td>
</tr>
</tbody>
</table>
## Verification on Various Inputs

- **Results of 10-fold cross-validation**

<table>
<thead>
<tr>
<th>Data Set (# of classes)</th>
<th>SSVM</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Training Error (%)</td>
<td>Test Error (%)</td>
</tr>
<tr>
<td>Handwriting (11 classes)</td>
<td></td>
<td>1.47</td>
<td>2.89</td>
</tr>
<tr>
<td>Mouse (8 classes)</td>
<td></td>
<td>6.82</td>
<td>8.79</td>
</tr>
<tr>
<td>Game (94 classes)</td>
<td></td>
<td>7.59</td>
<td>14.34</td>
</tr>
</tbody>
</table>
Recognition instead of Verification?

- Results of 10-fold cross-validation

<table>
<thead>
<tr>
<th>Data Set (# of classes)</th>
<th>SSVM Training Error (%)</th>
<th>SSVM Test Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Animal (3 classes)</td>
<td>5.10</td>
<td>8.75</td>
</tr>
</tbody>
</table>

TraClass: Trajectory Classification Using Hierarchical Region-Based and Trajectory-Based Clustering
Jae-Gil Lee, Jiawei Han, Xiaolei Li, Hector Gonzalez. Int. Conf. on Very Large Data Base VLDB, Aug. 2008.

(A) Animal (L5)  
(B) Animal (L100)
Conclusion

- We propose a trajectory-based approach for verification and recognition such as for detecting game bots.
- We propose a scheme with temporal information based on manifold learning and a novel dissimilarity measure.
- We tested our models on various input trajectories.
- How can we generalize our success to other input trajectories?
- How much can we work on recognition?
Thank You!

Hsing-Kuo Kenneth Pao
pao@mail.ntust.edu.tw