Identifying Game Players with Mouse Biometrics

Ryan Kaminsky, Miro Enev, Erik Andersen

December 8, 2008

Abstract

Recent work in mouse movement analysis has determined that, with sufficient data, users can be uniquely identified solely by their mouse movements. We consider the domain of video games and attempt to use mouse movements to identify game players. We conduct a user study that requires users to perform baseline tasks in a controlled environment, and then play Solitaire and StarCraft, two popular video games. We extract features from the mouse movement raw data and employ basic (k-Nearest Neighbor) and complex (Support Vector Machine (SVM)) machine learning techniques to classify users. The accuracy of the identification varies with the variety of moves available in a particular game, but we are able to identify the players in our user study with accuracy well above what random chance would predict.

1 Introduction

Authentication and privacy are important not only for secure systems like electronic banking and email, but also for video games. Currently, there are thousands of video games on the market. Many of them are casual games, such as Solitaire, Minesweeper, and online Flash games, while others are more serious, such as the popular real-time strategy game StarCraft, or the first-person shooter game Halo. In each of these games, players use the mouse and keyboard to interact with the computer and, in so doing, develop their own style of interaction. It is possible that this style is unique enough that it could leak information about the player, which is a concern from a privacy standpoint. On a more positive note, however, uniquely identifying players by their styles could lead to stronger authentication and make cheating more difficult.

Recent work in using machine learning to identify users by their mouse movements has produced some promising results. Thus, we present a system that records mouse movements while a player is playing a video game, analyzes this data, extracts features, and uses them to identify the player. We are able to uniquely identify players as well as determine a player's gender.

We have conducted a small user study to gather data on user interaction in a controlled baseline application and two video games: Solitaire and StarCraft. Solitaire is a simple, casual game, while StarCraft is a more complex game requiring more detailed mouse movements. The user study consisted of fifteen participants who played both games and performed some baseline tasks such as generic mouse moves, clicks, and drags.

For both games, we learned the user identification model using part of the user data and tested our model with the rest, using k-Nearest Neighbor (kNN) [5] and Support Vector Machine (SVM) [4] learning methods. Our data shows that game players can be uniquely identified using mouse movements under certain conditions.

2 Related Work

Modern GUI work is based in part on Fitt's law [7] and the Accot-Zhai steering law [1]. This is especially true regarding mouse movement analysis. These laws relate the time required to perform a task, using a pointing device like a mouse, with other attributes of the task, such as distance traveled. Fitt's law models human behavior by creating a relation that predicts the time necessary to select a target during a series of rapid moves as a function of the distance and size of the target. The Accot-Zhai steering law is a 2D version of Fitt's law. We use these in the construction of our baseline application.

Weiss et. al. [12] studied mouse movement biometrics by having users perform tasks such as button presses and mouse drags for a fixed pattern on a standalone application that gathers data. They then analyzed the mouse movement data, created feature vectors and attempted to identify individual users using k-NN learning techniques. Our work uses a similar baseline application, but we also capture mouse movement data for two games, both having more random mouse movements than the baseline. We also have over double the number of study participants. We think the increased number of participants and the randomness introduced by complex games adds additional complexity to the problem.

Ahmed and Traore [3] attempt to detect unauthorized access using mouse and keyboard dynamics and construct a Mouse Dynamic Signature (MDS) of seven factors for each individual. Pusara and Brodley [9] attempt to detect changes in mouse movement biometrics and query users to reauthenticate to verify their session hasn't been hijacked. A survey of authentication methods based on mouse movement biometrics is found in [11]. Our work differs from these works in that they focus on authentication of users in a controlled setting. Mouse use in video games is more dynamic and unconstrained and poses a more challenging problem.

3 Data Collection

We conducted a user study to obtain game player mouse-movement data. The study consisted of three parts: performing the baseline experiments, playing Solitaire, and playing StarCraft. The participants were all graduate students in Computer Science at the University of Washington. Ten of the participants were male and five were female. All of the participants were familiar with the rules of Solitaire and had played the game before. Eleven of the participants knew how to play StarCraft, and we gave a short tutorial to the other four participants who had not played before. All four of these beginners were female. We ran all of our experiments on the same computer, to ensure that parameters such as processor speed, mouse sensitivity, monitor size, and resolution were consistent for all users.

A natural extension to our framework would be the addition of keyboard feature analysis, however, we chose to focus exclusively on mouse events as a means of user identification for

several key reasons. Firstly, we are interested in applying our methods in the web domain where user interactions are predominantly generated by pointing devices such as mice and pen-tablets. Furthermore, in a web context keyboard events are only induced by form-based sites or identification/password entry fields and keystroke dynamics analysis has already been applied in these settings to provide stronger authentication [14]. Lastly, the practical success of keystroke dynamics [2] leads us to believe that appending keyboard features to our machine-learning methods would most likely improve our results.

3.1 Baseline experiments

We developed an application to capture baseline mouse actions from the users in our study. This application allowed us to gather data in a controlled environment for each individual so that we could compare it with his or her mouse action characteristics gathered from playing the games. Additionally, it allowed us to explore higher-level mouse events, such as clicks and drags, to find mouse characteristics that could identify an individual user with more accuracy.

The baseline application consists of three major areas that attempt to capture three of the most important mouse actions that an individual can perform: mouse moves, clicks, and drags. Screenshots of each of the tasks appear in Figure 1. The first task requires that the user click as rapidly and accurately as possible between two targets that are evenly spaced on each side of the window's center point. We ensure that both the horizontal and vertical directions are covered and vary the distances and target sizes. The second task requires users to drag a circular shape to a target location from sampled angles covering a 360 degree range. The final task requires users to double click on a target. By providing specific, detailed tasks, we constrain the degrees of freedom of mouse actions and can compare individual users' actions more easily. The baseline application collects general mouse movement data as well as accuracy statistics such as distance from the center of a target.

3.2 Solitaire

The second phase of our user study involved playing Solitaire, a single-player, casual video game that involves the movement of virtual playing cards. The objective of the game is to move cards from a shuffled deck onto a tabletop and into piles that are sorted by suit. The cards must be moved in a way that adheres to a set of rules. The game is freely distributed with many versions of Windows. As a result, Solitaire is quite common and many people have some experience playing the game.

We used the standard Solitaire application that comes with Windows XP for our user study. The particular arrangement of cards in the game was chosen at random. We instructed users to play for five minutes, and asked them to start a new game if they won or became stuck.

3.3 StarCraft

The final phase of our user study involved playing StarCraft. StarCraft is a real-time strategy game that was released for the PC by Blizzard Entertainment in 1998. Players build small

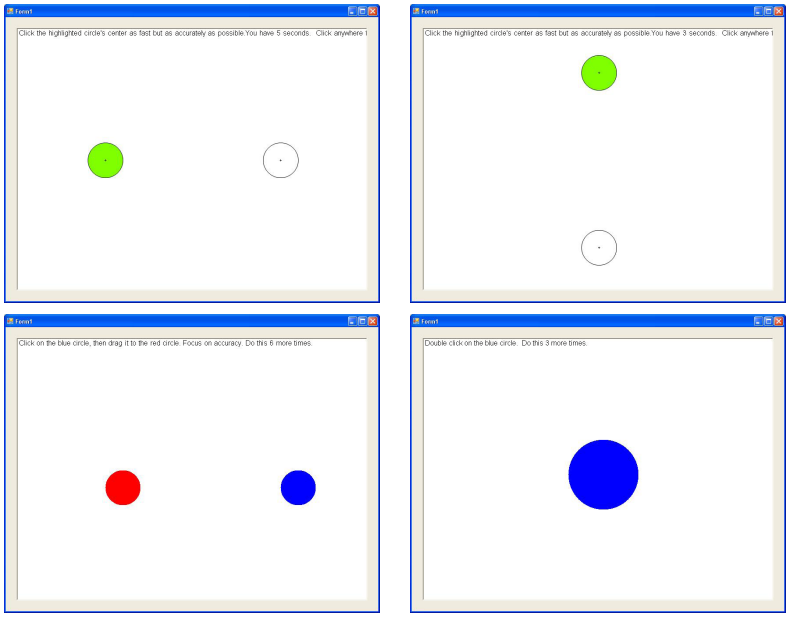

Figure 1: Screenshots of the baseline program. Top left and top right: clicking tasks. Bottom left: dragging tasks . Bottom right: double clicking tasks.

characters, move them across a playing field, and use them to attack the armies of other players. Timing is a critical concern, and playing competitively requires a large number of mouse movements. StarCraft is not as ubiquitous as Solitare, but retains a devoted fanbase and is played in online tournaments.

In our study, each user played against a single AI-controlled opponent on the same two-player map, which is named Astral Balance and is at the top of the list of Blizzard-constructed maps. Each of the players played as the Terran race, but the race of the opponent was randomly chosen to be either Terran, Protoss, or Zerg. The user's starting location in the map was either at the lower-left corner or the upper-right corner, with equal probability.

Each user played StarCraft for fifteen minutes. If the player won or lost before fifteen minutes had elapsed, we restarted the game and continued recording user data. As a wide variety of skill levels were represented by the participants, the specific games had varying outcomes. In some games, players finished the game in a strong position, and in others, players finished the game in a weak position. This adds an additional challenge to the classification problem, as user interaction could vary depending on the result of the game.

4 Analysis and Results

4.1 Event logging and action extraction

We developed a C# program that adds hooks to the Windows API to log low-level mouse events. We use this program to capture mouse movements, left and right mouse button presses, and left and right mouse button releases. Using MATLAB we parse these raw events

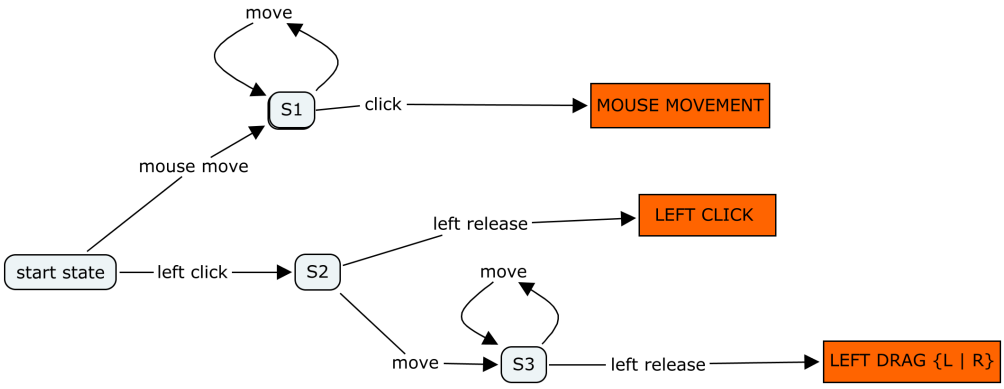

Figure 2: State machine logic for parsing events into actions.

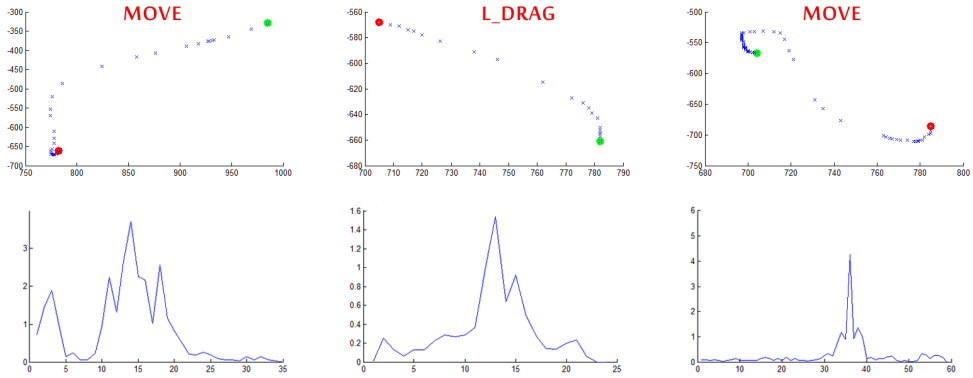

Figure 3: Visualizations of the first three movements in a solitaire game. Top row: each "x" represents the (x, y) pixel location of a mouse event comprising the action. The red and green circles represent begin and end events. Bottom row: the normalized velocity for each mouse event for the action above in pixels/microsecond over the course of the action.

into three composite actions: mouse movements, clicks, and drags. We do this with the state machine logic of Figure 2. An example of higher level actions and statistics constructed during a game of Solitaire is shown in Figure 3. We use the key-logger program and state machine logic to record and process mouse events for baseline tasks and both games.

4.2 Baselines

We generate feature vectors (Section 4.4) from the baseline data and then run several experiments training and testing on this data (Section 5). We also examine this data for the tradeoff between accuracy and speed predicted by Fitt's Law:

$$T = a + b \log_2(\frac{D}{W} + 1) \tag{1}$$

In this equation T is the average time for the task, a and b are experimentally determined

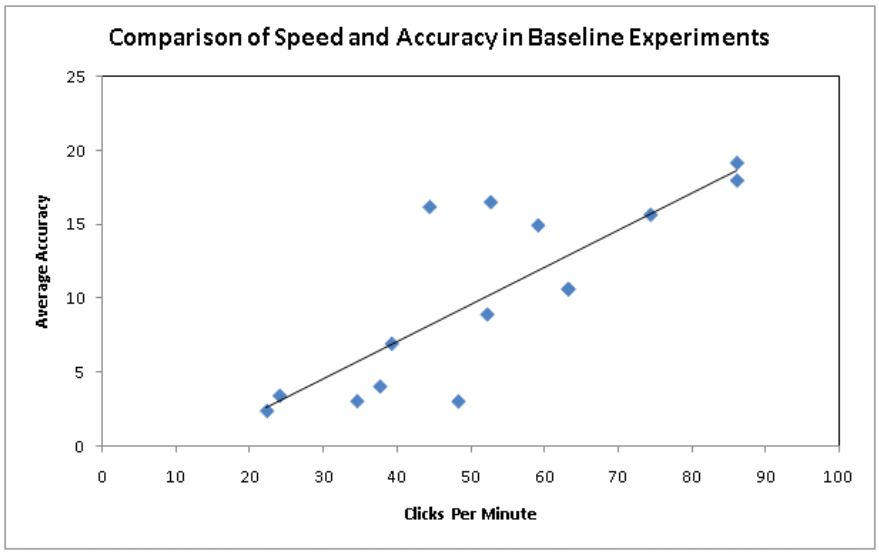

Figure 4: Graphical comparison demonstrating the tradeoff between speed and accuracy as demonstrated by Fitt's law.

constants for a particular device, D is the distance to the center of the target and W is the width of the target. We see that there is an direct relationship between the distance D of the target and the time T of the task, and an inverse relationship between the width W of the target and the time T. This gives rise to the tradeoff between speed and accuracy. Figure 4 demonstrates the relationship between these two attributes for a pointing task, using a mouse to click on a target, from our baselines. The relationship is linear because in this case, a lower accuracy number translates to *better* accuracy.

4.3 Solitaire and Starcraft

Example data from three users is shown in Table 5. The variability in the table is representative of the subject pool. Note that the inter-user differences in the feature vectors of StarCraft are much greater than the variability in Solitaire. This phenomenon could be attributed to several factors. One such factor is that there were more pronounced differences between novices and experts in StarCraft. Another possible explanation is that the action space in Solitaire essentially restricts the user's controls to dragging cards of fixed dimension to one of a few possible locations; in StarCraft the controls are much more fluid and allow the user a few different ways to achieve a particular in-game task.

4.4 Features

Based on the collected raw data, we created several features that help to uniquely identify users. For all mouse moves without a button press (mouse move), mouse drags with the left button pressed (left drag) and mouse drags with the right button pressed (right drag) we compute the features below:

- *Path Length Mean* - The average path length of all action types of the same class. The path length of a single action is defined as in [12] $\sum_{i=2}^{n} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}$
- *Path Length Standard Deviation* - The standard deviation of the path length of all actions of the same class.
- *Velocity Mean* - The average velocity of all actions of the same class. The velocity of a single action is defined as in [12] $\frac{1}{n} \sum_{i=2}^{n} \frac{\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}}{t_i - t_{i-1}}$
- *Velocity Standard Deviation* - The standard deviation of the velocity of all actions of the same class.

For left and right mouse button click actions we compute the following features:

- *Click Length Mean* - The average length of time of all clicks. The click length is defined as the difference in timestamps between the button down and button up events.
- *Clicks per Minute* - The number of clicks per minute in a particular time slice.

As a final feature relating to clicks, we compute the *Total Clicks per Minute* as the sum of *Clicks per Minute* for the right and left mouse buttons. We also create features based on the spatial information generated by a user's session:

- *Percentage of Clicks in Quadrant* - We divided the screen into four logical quadrants: Northwest, Northeast, Southwest and Southeast and compute the percentage of clicks occurring in each quadrant.

The total vector for each user initially consisted of 21 features. Through empirical experiments we determined that some features adversely affected the accuracy of user identification. We determined that the four features comprising the *Percentage of Clicks in Quadrant* were too specific to the game and the particular minute of the game. For example, early in StarCraft, a player may spend a large amount of time collecting resources in the Northeast portion of the screen, returning here only sporadically during the remainder of the game. Such a pattern was common in the data and invalidates these features for the purpose of identification. Therefore, the final feature vector includes 17 features.

4.5 Feature information gain

The Weka [13] toolkit offers tools for determining which attributes provide the most information gain [8] for a particular classification task. Table 1 displays the top three features in terms of information gain for each recognition task. *Left Click Length Mean* is the best indicator when recognizing individuals in both Solitaire and StarCraft. Gender differences for StarCraft become highly visible in the number of *Clicks per Minute*, while *Left Drag Velocity Mean* is the best indicator of gender for Solitaire. These results clearly show that there are distinct mouse movement characteristics for both individuals and genders, but that the particular distinguishing characteristics vary across games.

	User 1	User 2	User 3
Solitaire			
Mouse Move			
Vel Mean	0.29927	0.41428	0.39501
Vel Std Dev	0.32838	0.54221	0.43745
Path Length Mean	348.7095	248.0512	213.8354
Path Length Std Dev	501.3774	270.3284	250.7962
Clicks			
Left/Min	18.0847	16.2154	35.0093
Left Length Mean	0.18744	0.11744	0.18469
StarCraft			
Mouse Move			
Vel Mean	0.53084	0.70551	0.77778
Vel Std Dev	0.6316	0.92241	1.1966
Path Length Mean	1158.0631	206.9399	603.4157
Path Length Std Dev	1548.9047	306.7001	907.565
Clicks			
Left/Min	5.0026	48.1123	20.0625
Left Length Mean	0.1938	0.081396	0.0844
Right/Min	13.0067	65.152	18.0563
Right Length Mean	0.15008	0.069769	0.0825

Figure 5: Sample of user features gathered during our study.

Rank	Solitaire		StarCraft	
	Individual	Gender	Individual	Gender
1	Left Click Length Mean	Left Drag Velocity Mean	Left Click Length Mean	Total Clicks/Min
2	Total Clicks/Min	Left Drag Velocity Std Dev	Right Click Length	Left Clicks/Min
3	Left Clicks/Min	Right Drag Path Std Dev	Mouse Move Velocity Std Dev	Right Clicks/Min

Table 1: Features with the most information gain for each game and recognition task.

5 Results

We first attempt to learn models for each game individually. To increase the amount of training data, we split the Starcraft and Solitaire event data streams into one-minute segments and learn SVM, 1-Nearest Neighbor, and 7-Nearest Neighbor models. We used RapidMiner [10], a free program downloadable from the Internet, to construct our SVMs. There are several adjustable parameters available on the SVM. We determined the most effective parameters empirically and used the same parameters for all datasets. For the nearest neighbor algorithms we also used the downloadable Weka [13] toolkit. The L-1 Norm is used for calculating nearest-neighbor distance and for the 7-Nearest Neighbor test runs, we weighted each neighbor's vote by the inverse of its distance from the test feature vector.

For testing, we attempt to identify each individual via ten-fold cross-validation. The left side of Table 2 summarizes our results. In all of these examples, the chance of correctly

Game	Individual		Gender	
	Model Type	Accuracy	Model Type	Accuracy
Solitaire	SVM	57.5%	SVM	86.8%
	1NN	48.0%	1NN	78.7%
	7NN	49.3%	7NN	81.3%
	Random Chance	6.7%	Random Chance	66.7%
StarCraft	SVM	79.3%	SVM	93.3%
	1NN	63.9%	1NN	89.7%
	7NN	65.5%	7NN	93.8%
	Random Chance	6.7%	Random Chance	66.7%

Table 2: Accuracy at identifying individuals and gender with cross-validation.

identifying a particular player by picking randomly is 1/15, or 6.7%, yet we are able to identify people with much higher accuracy—57.5% for Solitaire and 79.3% for StarCraft. Note that the SVM performs better than both nearest neighbor methods. In neither case are our models able to identify players perfectly, but we believe our data shows that mouse movements do indeed leak a user fingerprint that can be used to identify players. We also point out that the accuracy for StarCraft is higher than Solitaire. We believe this refutes our hypothesis that the greater variety of motions that a player is expected to perform in StarCraft would make classification more difficult.

Given that our user study contained 10 men and 5 women, we now try determine if gender plays a role in the uniqueness of an individual's mouse movements. Our results can be seen in right side of Table 2. On this easier task, our models perform much better percentage-wise, identifying 86.8% of gender classes for Solitaire and 93.3% for StarCraft. However, the random chance classification is also much higher at 66.7% if the classifier chooses male for all subjects. Note again that the SVM outperforms nearest neighbor classification.

We observe that our user study has a limited number of people, and therefore our sets of men and women are probably not representative of the sets of all men and women. We may have simply reduced the classification task to identifying whether a particular user is more likely to be part of one group that happens to include ten men or another group which happens to include five women. Nevertheless, our models seem to perform reasonably well on this task.

In the case of StarCraft, there was a significant difference between the male and female participants in terms of experience. All of the men were familiar with the game, while only one of the females was. Therefore, it could be that our analysis of gender for StarCraft is actually more related to skill than gender. However, there was no experience gap in Solitaire, and we are able to achieve similar results. This suggests that while the StarCraft gender model may be leveraging differences in skill as well as gender, our model also seems to work when the skill of the players is similar.

We were particularly interested to see if our models we able to identify users across games. For example, we wanted to know if a model learned on Solitaire could identify users playing StarCraft. Unfortunately, our models were not so successful. Table 3 shows our results. These results seem to suggest that our features are primarily capturing game-

Model Learned On	Model Applied To	Model Type	Accuracy
Solitaire	StarCraft	SVM	13.9%
		1NN	12.4%
		7NN	13.4%
		Random Chance	6.7%
StarCraft	Solitaire	SVM	26.7%
		1NN	17.3%
		7NN	12.7%
		Random Chance	6.7%

Table 3: Accuracy at identifying users across games.

specific fingerprints. This is disheartening, perhaps, although it does not necessarily mean that there are no features that would work across games.

Finally, we analyzed our baseline tasks. We computed one feature vector for each individual, learned our models, and then applied these models to identify users in Solitaire and StarCraft. Conversely, we also attempted to classify user baselines by applying models learned on each game to the baseline feature vectors. Our results can be seen in Table 4. It appears as though neither direction was successful. Our original goal when using baselines was to isolate individual motions by reducing the variability of movements a user performed. We believed this would allow us to capture canonical mouse characteristics of individual users that would be easily identifiable as fingerprints when a user plays a game. Additionally, we feared that extracting this fingerprint from game data would be too challenging given the unconstrained mouse movement space. Essentially, we felt that baseline data would give a "clean" fingerprint, whereas game data would be too cluttered.

However, it turns out that the models from the games themselves are sufficient for identification because of the large amount of data and the power of the classification tools. It appears as though our baseline tasks did not sufficiently capture the fingerprint space of an individual. We hypothesize that we did not capture enough baseline data to build a model that is powerful enough to be sensitive to differences in user interaction. Instead, simply having users perform many actions within a game and building the model from game input data was more successful.

Given that mouse usage provides identifiable information, we recognize that it could be susceptible to forgery as with many other authentication techniques. However, simply capturing and reconstructing someone's mouse events would only be useful for accomplishing the tasks that the user was originally trying to perform. Any variation on the specific task would make it difficult to artificially recreate a sequence of mouse moves with the proper signature that successfully completes the new task.

6 Applications

There are a number of applications where the ability to identify an individual based only on mouse movements would be valuable both within the realm of video games and within a

Model Learned On	Model Applied To	Model Type	Accuracy
Baseline	Solitaire	SVM	10.7%
		1NN	12.0%
		7NN	12.0%
		Random Chance	6.7%
Baseline	StarCraft	SVM	6.2%
		1NN	6.7%
		7NN	6.7%
		Random Chance	6.7%
Solitaire	Baseline	SVM	20.0%
		1NN	20.0%
		7NN	20.0%
		Random Chance	6.7%
StarCraft	Baseline	SVM	20.0%
		1NN	13.3%
		7NN	20.0%
		Random Chance	6.7%

Table 4: Accuracy at identifying users with baselines.

broader context. With massively-multiplayer online games becoming more popular, people are seeking to make money both through game competitions and gambling. Profit-making provides an incentive for attackers to game the system through many means. For example, if someone hijacks a game player's account, the attacker could purposely reduce the legitimate user's score or ranking, or transfer money or objects of real world value. In online poker, a hacker, having logged into someone's else account, could purposely lose several hands to transfer money to a co-conspirator. By using mouse metric signatures, a different user could be detected and the original user alerted. This could also be applicable to many online websites, banking sites for example, where a hacked account can prove costly. Although contextually different from a video game, a banking site could ask a user perform a mouse task that is able to capture signature information as well as a video game.

Another application involves identifying cheating players. In StarCraft, there are several hacks available that can give advantages to one party over another. One such hack is called a "map hack" and allows a player to see what the other players are doing at all times. One approach for dealing with this problem is to ban cheating players from online servers, but they can simply log back on with different account names. By using mouse signatures, online server operators could potentially identify repeat offenders and ban them more effectively. This approach could be extended further to detect bots. To gather the required level of mouse event granularity, nearly any client (application, webpage, etc.) could be instrumented to record this data.

There are also potential privacy concerns associated with such a mouse tracking system. For example, public computers installed with such software could be used to track users' movements around a city or time usage patterns (person A uses this computer from 10-11am on Mondays and Wednesdays).

7 Future Work

One area of future work is to build a model that can work across many game domains. We had hoped that our baseline analysis and feature selection would allow us to build such a model, but this turned out not to be the case. A more thorough analysis of how a user's interaction style changes based on his or her specific activities could help develop a set of features that account for these differences, leading to a more accurate model.

Our experiments were conducted in a controlled setting, using the same computer, mouse, display and resolution. We wonder if experiments under less ideal conditions, involving different mice, computers (desktops vs laptops) and resolutions would yield similar mouse signatures. Additionally, we would like to explore the possibility of identifying users on the web where the mouse is the input device of choice. In this setting we do not have direct access to operating system mouse events so we would have to rely on capturing JavaScript mouse events. Using the web may also allow us to examine if our techniques scale to hundreds and thousands of uses or if a different approach is needed.

Another area of exploration involves anonymizing mouse movements. In the same way that network traffic is anonymized using systems such as Tor, developed by Dingledine et. al. [6], a mouse anonymization system could add artificial actions and delays into the mouse data stream to anonymize a signature.

8 Conclusion

We present a system for identifying individual users based on mouse movement dynamics. We identify users in three distinct settings: a baseline application with controlled movements, a casual game with slightly less controlled movements and a full screen action game with random movements that change based on players and the game scenario. Using mouse movement data that we collect during a user study, we develop features that are unique to individual players and that we use to identify them in a set of test data. Our system is able to uniquely identify users with an accuracy rate much higher than random chance, but the models we constructed do not perform well across game domains. However, we believe it is accurate enough to be useful for applications that attempt to identify cheating players or unauthorized users.

References

[1] J. Accot and S. Zhai. Refining fitt's law models for bivariate pointing. In *ACM CHI Conference on Human Factors in Computing Systemsm*, pages 193–200, 2003.

[2] Admit one security. `http://www.admitonesecurity.com/`.

[3] A. A. E. Ahmed and I. Traore. Detecting computer intrusions using behavioral biometrics. *Privacy, Security, and Trust*, 2005.

[4] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20, 1995.

[5] T. Cover and P. Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27, Jan 1967.

[6] R. Dingledine, N. Mathewson, and P. Syverson. Tor: the second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium*, volume 13, August 2004.

[7] P. M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47(6):381–391, 1954.

[8] T. M. Mitchell. *Machine Learning*. The Mc-Graw-Hill Companies, 1997.

[9] M. Pusara and C. Brodley. User re-authentication via mouse movements. 2003.

[10] Rapidminer. `http://rapid-i.com/`.

[11] K. Revett, H. Jahankhani, S. T. de Magalhes, and H. M. Santos. A survey of user authentication based on mouse dynamics. In *Communications in Computer and Information Science Global E-Security 4th International Conference*, June 2008.

[12] A. Weiss, A. Ramapanicker, P. Shah, S. Noble, and L. Immohr. Mouse movements biometric identification: A feasibility study. In *Proc. Student/Faculty Research Day, CSIS, Pace University*, pages 1–8, May 2007.

[13] Weka. `http://www.cs.waikato.ac.nz/~ml/weka/`.

[14] Wikipedia entry on keystroke dynamics. `http://en.wikipedia.org/wiki/Keystroke_dynamics`.