Handling Artificial Acceleration in Mouse Movement Biometrics

Shawn C. Gross and John V. Monaco
Seidenberg School of CSIS, Pace University, White Plains, New York

Abstract
The purpose of the research is to focus on the possible usage of the Mouse Movement Biometric system for recognition of users. The data that was used was gathered through a web-browser based program, which recorded multiple sessions for all 39 users. The study focuses on analyzing and solving various problems that exist when attempting to use this data for pattern recognition and for normalization. The majority of this study revolves around the main problem of artificial acceleration in mouse movements contained within the Mac OS X and Windows operating systems. Included are descriptions of this problem and possible solutions that will allow for further research into pattern recognition.

Index Terms – Authentication, Biometrics, Artificial Acceleration, Mouse Threshold, Python

1. Introduction
During the current age, the usage of computers on a daily basis has grown from a luxury to a necessity. We use computing devices nearly every hour out of every day whether it is through a smartphone, tablet, laptop, or even a traditional desktop. Alongside this ever growing prevalence of technology is the need for constant evolution and research in the field of computer security. From malware to viruses, wherever a secure network or a database containing sensitive information exists, security threats are sure to follow. The traditional means of dealing with personal security have mostly been centered on the username and password method.

The main problem with this archaic method is that after a password is given into a secure system, a malicious user may never be challenged again in any other manner as they have access to the information, systems, or applications which lie behind the layer of security already bypassed. Even if this malicious user were challenged again, they would already have had access to the password, resulting in an overall weak method of security. In order to confront this problem, the introduction of biometrics into computer security has begun to gain popularity.

Biometrics in computer security are considered recognition or authentication methods, based off of human characteristics or behaviors, for users of a secure system. The introduction of biometrics into this field is to obtain an authentication method that would render the username/password method obsolete as well as adopt a security standard that would guarantee absolute security by gauging features that differ entirely from user to user.

Most recently, certain biometric measurements such as voice, iris, and fingerprints have gained popularity by being implemented on devices such as laptops, tablets and smartphones. When enabled, these measurements ensure that only the user will be granted continual access throughout the course of the session and will minimize any unauthorized access. While it has proven that these measurements in the field of biometrics have proven to be effective, more research is being conducted into mouse movements and how they can be used in a similar manner.

If mouse movements can be proven to be as effective as iris or fingerprint methods, a powerful system of multiple biometric authentication methods could be introduced. With such a security system, continual security checks through the user’s session would ensure security that is exponentially more secure over a traditional password security system.

Expanding on the ongoing mouse movement biometric research at Pace University, the research contained in this study will help to solve a problem that will allow for more rigorous research into this biometric field. This paper is structured in the following manner: Section 2 describes the problem of artificial acceleration/ Sections 3&4 describe how artificial acceleration can be bypassed. Section 5 describes the research conducted on pre-gathered data, affected by artificial acceleration, and how the application of mathematics can attempt to decipher the true velocity and mouse movements. Section 5 will describe the results of the research applied in Section 3. Section 5 will include conclusions, problems encountered and recommendations on future research.

2. Past Biometrics Research
At Pace University, mouse movement biometric research has been conducted since 2007. The first part of research consisted of an analysis of a currently existing mouse movement biometric system. Emphasis was placed on details of this system, explaining how key features for user recognition are extracted from the system.
The focus for this research was placed on mouse curvature calculations as well as recorded click events however, acceleration was only mentioned and not a consideration. Similar research was again conducted in 2013 and 2014, suggesting more focus on feature extraction, and delving further into mouse trajectory features.

In 2014, the design of the mouse movement biometric system used for identifying participants of quizzes was created. This system used the foundations of the prior research of students at Pace University to create a system to collect data on various users with the intent of using this data to recognize their patterns, ultimately being able to distinguish the users based off of their mouse data. The system used the basic mouse trajectory features to characterize the mouse movement recorded during each quiz session. A brief description of these features are listed below:

1. **Number of Trajectory Points:**
   The total number of \((x,y)\) coordinates in a curve
   \[
   \sum_{i=1}^{n} (p_i)
   \]

2. **Amount of Time to Complete Trajectory:**
   Measures the time of trajectory from start to finish
   \[
   \sum_{i=2}^{n} (t_i - t_{i-1})
   \]

3. **Length of Trajectory:**
   Measures the length by calculating the distance between each point in trajectory.
   \[
   \sum_{i=2}^{n} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}
   \]

4. **Velocity from Point to Point in Trajectory:**
   Calculates the velocity between each point in the Trajectory.
   \[
   V_i = \frac{\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}}{t_i - t_{i-1}}
   \]

5. **Acceleration from Point to Point in Trajectory:**
   Uses the velocity between two points and the time between the movement to calculate the acceleration.
   \[
   a_i = \frac{V_i - V_{i-1}}{(t_i - t_{i-1})/2}
   \]

6. **Direction Angle from Point to Point in Trajectory:**
   Measures the angle:
   \[
   m_i = \frac{y_i - y_{i-1}}{x_i - x_{i-1}}
   \]
   Then the slope:
   \[
   m_i - m_{i-1}
   \]

7. **Number of Inflection Points in the Trajectory:**
   Number of changes in the curves of the movement.

As noted in this research, while describing these features, considerations were made to the sampling rate, screen resolution and operating system. These variables, when not considered, can skew experimental research results. While this system was created, the problem of artificial acceleration arose. Along with monitor refresh rate, screen resolution, and operating system, artificial acceleration is the increase in velocity after the physical movement of the mouse crosses a defined threshold value, stored in the operating systems registry or defaults. While the above features allowed the mouse movement biometric system to capture data in an efficient manner, without considerations towards artificial acceleration when a user rapidly moves their mouse, the calculations achieved by these features will not reflect accuracy, leading to false results when using mouse pointer movement to calculate physical mouse movements.

### 3. Artificial Acceleration

Back in 1999, engineers at Microsoft were busy creating Windows XP under the guise of “Whistler”. While the engineering for XP was underway, they decided to address a problem that many users of prior Windows operating systems had complained about. This problem was of a sluggish mouse pointer that could not handle fast scroll speeds coupled with ever increasing monitor resolutions, confusion amongst users who were not familiar with acceleration, and the old pointer ballistics from earlier
The systems applying separate acceleration algorithms to the x and y-axis resulting in imperfect mouse movements. They confronted these problems by introducing artificial acceleration, an algorithmically applied function that will increase the velocity of the mouse pointer based off of the velocity of the movement of the mouse which is controlled by the user’s hand. This results in a much more responsive pointer when dealing with increased mouse velocities, eliminating the slow mouse pointer velocity of earlier operating systems.

Before XP, velocity was handled by the simple velocity bar that is shown in figure 2.1. When you move this sliding scroll bar, it would automatically default to pre-existing tick marks, hidden from the user. At the slowest default speed, scroll bar extended to the left furthest position, the resulting velocity is a 1:1 ratio between the mouse and the pointer’s movement. If the mouse moves at a velocity of 1 inch/s, then the pointer is also moving at the same, slow velocity. As the mouse scroll speed bar is increased, the pointer velocity is increased in a corresponding ratio to the mouse movement.

In XP and future Windows Operating Systems, the introduction of a transfer function to calculate artificial acceleration was created which would be algorithmically applied to the pointer. Before the transfer function could be created, the arbitrary units of the mouse had to be converted to physical units or in other words, the arbitrary units or the mouse and of the pointer were being converted to the actual velocity of the mouse and the pointer we see on the monitor. Figure 2.2 depicts the mathematical formulas used for this conversion.

\[ V_{\text{mouse}} = \text{mickey} * \frac{\text{MouseBusUpdateRate}}{\text{PointerResolution}} \]

**Figure 2.2. Mouse conversion formula [1]**

In both of these formulas, ‘mickey’ is representative of the number of coordinate changes per mouse packet. With these two formulas, we can establish a ratio between the velocity of the pointer and of the actual mouse movements. In practice, this ratio turns out to be a gain dependent on the amount of arbitrary coordinates that are given per mouse packet generated. For example, three coordinates given in a packet would result in a gain of 3 meaning that the pointer would move with a velocity that is 3 times faster than the mouse. By default, USB mice have a packet generation rate of 100 packets per second while a mouse is moved. [2] These formulas will be useful for future data collection as long as that collection includes a recording of the screen resolution as well as the refresh rate of the monitor and artificial acceleration is in fact, disabled.

It is also important to note that artificial acceleration is enabled through the mouse options but it is also influenced by the pointer speed as depicted through the four speed curves in Figure 2.4.

**Figure 3.3. Pointer conversion formula [1]**

\[ V_{\text{pointer}} = \text{mickey} * \frac{\text{ScreenUpdateRate}}{\text{ScreenResolution}} \]

**Figure 2.4. Artificial Acceleration Velocity curves [2].**

Normally the velocity of the mouse is dependent on the two previously noted formulas and the amount of coordinates per packet received but here at each point on the very curve, we notice an inflection resulting in a great pointer speed being applied. This point is the artificial acceleration increasing the velocity of the pointer at a rate that is doubled that of velocity before the points. Normally velocity would carry on in a linear manner without such severity as witnessed by these inflection points. These points are actually threshold values that reside in the Windows registry. They determine at what mouse speed in inches/second the acceleration, of the
pointer, should be doubled for the first value, as long as the registry setting for the variable MouseSpeed is set to 1 or 2, and quadrupled, as long as the registry setting for the variable MouseSpeed is set 2 for the second value. So with artificial acceleration applied, we can see at what mouse speeds, as denoted in the registry, acceleration will be applied outside of the normal ratio which is coordinates per second. IE three coordinate changes per one data packet measured in seconds would equal a ratio of three so the velocity of the pointer is roughly three times more than the velocity of the mouse, the three being dependent on how many coordinate changes occurred.

4. Disabling in Windows Operating Systems

In the Windows Operating systems succeeding Windows 2000, artificial acceleration is enabled by default. This can be circumvented manually with relative ease by navigating to Control Panel / Hardware and Sound / Mouse (underneath Devices and Printers) / Pointer Options uncheck the Enhance Pointer precision box. Figure 3.1 shows the Mouse Properties pane in Windows 7

![Figure 3.1 Mouse Properties Pane](image)

As witnessed, the only changes now to the pointer are the basic pointer speed multiplier based off of the pointer speed bar just as noted previously in older Windows operating systems. An alternative manner in which the acceleration can be disabled is through the Windows registry. The Windows registry can be accessed through the Registry Editor. In the start menu type in regedit and click on the Registry Editor. Navigate to HKEY_CURRENT_USER/Control Panel/Mouse and note the following fields:

- **MouseSpeed**: Determines the speed of the pointer dependent on the speed of the mouse.
- **MouseThreshold1**: Acts as a trigger for when artificial acceleration is applied to the speed of the pointer whose speed is doubled, as long as MouseSpeed has a value of 1 or 2.
- **MouseThreshold2**: Acts as the second trigger for when artificial acceleration is applied and the speed of the pointer is quadrupled, as long as MouseSpeed has a value of 2.

It is of importance to note that we can effectively manipulate the artificial acceleration as well as disable it from this registry edit. Figure 3.2 shows the current registry edits to these three fields.

![Figure 4.2 Registry Editor](image)

5. Disabling in Mac OS X

In every distribution of Mac OS X, artificial acceleration is handled in a similar manner to the Windows operating systems. Disabling artificial acceleration in OS X can be achieved through a simple Linux command (Figure 4.1) through the Terminal application.

```
defaults write .GlobalPreferences com.apple.mouse.scaling -1
```

![Figure 5.4 Disable acceleration command](image)

6. Experimentation on Pre-gathered data

At the beginning of this study, pre-gathered data was provided in the form of recorded sessions on student-users
who were taking quizzes for a class. The events as well as movement were captured and parsed into a comma separated value file or csv file (extension .csv). This csv file included the pertinent information of the x & y coordinates, as well as the time elapsed with each event. Figure 5.1 depicts an excerpt of the data provided in the csv file that was modified to reflect pertinent data.

**Figure 6.5. CSV file contents**

Given this information, the next logical step was to parse through this csv data and calculate the speed in between each recorded event. Certain user’s data had no lapse in the time suggesting an error in the script due to a bug but the majority of user’s data could be parsed and utilized in producing a speed calculation.

The first calculation was to determine the distance as depicted in Figure 5.2. XDiff represents the delta of the x coordinates in between each movement as yDiff does the same for the y coordinates.

\[
Distance = \sqrt{(xDiff^2 + yDiff^2)}
\]

**Figure 6.2. Distance formula for coordinates**

With the distance calculated, the value obtained could be used to further calculate the speed in between each event change. This was achieved by the usage of the formula contained in Figure 5.3. Time Difference is once again representative of the difference in between the recorded times of the current event and the time of the prior event.

\[
Speed = \frac{Distance}{TimeDifference}
\]

**Figure 6.3. Speed formula**

The result from this parsing script gave the distance traveled as well as the velocity in between each movement. Sample output from the parsing script is shown in Figure 5.4.

**Figure 6.4. Python parsing script sample output**

The results from this parsing script could be used to create visual models through pyplot or a similar python library. With these results however, no way to ever determine the user’s screen resolution or refresh rate of their monitor is possible since these results have already been gathered so the only practical purpose they can be used for is the creation of visual models in future research that could possibly be used to map the physical movement of the pointer and comparing similar mappings for similar movements.

7. **Experimentation – Python Prototype**

In order to experiment with artificial acceleration, a mouse tracking simulator was created as a test environment. This simulator takes into account the variables that were discovered from the efforts of prior biometrics research at Pace University. These variables are: monitor resolution, DPI, refresh rate of the monitor, and artificial acceleration. As the program is executed, the user is prompted for a user number as well as their monitor size in inches. A small report is generated as the program conducts a background query on the operating system, retrieving information pertinent to the system as well as values that are necessary when accounting for any instances of artificial acceleration.

**Figure 7.1 User prompted for information**

The monitor size is used to calculate the monitor’s dots per inch or DPI. Additional information such as, the screen width and height is queried from the operating system through the Python programming language. This is achieved through the Tkinter library on Mac OS X and the
The DPI is calculated as follows:

1. **Calculate the diagonal resolution:**
   Where W denotes the width of the screen and H denotes the height of the screen.
   \[ D_{Res} = \sqrt{W^2 + H^2} \]

2. **Calculate the DPI:**
   Where monitor denotes the queried monitor size in inches
   \[ DPI = \frac{D_{Res}}{\text{Monitor}} \]

The simulator creates a background window cropped to fit the size of the monitor, mimicking a fully maximized web browsing window.

**Figure 7.2 Mouse Simulation Environment**

This grey screen captures any mouse cursor movements and records the time of the movements as well as positional coordinates into a file. This recording is achieved by a method being called every time the mouse is moved, thereby creating a recording loop that is terminated when the simulation window is closed or the quit button is pressed. Upon conclusion of the recording, a comma separated value file, CSV file, is generated. An excerpt of this file is included in Figure 7.3.

**Figure 7.3 CSV Results from Simulator**

Based off of the information returned by the operating system query, the appropriate values are recorded into the rows. This information is now prepared for the second phase of the program that includes the calculation of the mouse movements as well as a plot of the session.

Analysis was conducted by reading in this CSV file and pushing the values into a set of lists. The X and Y coordinate values were used for the generation of the scatter/line plot of the session while the Time list was used to generate the number of “mickeys” or coordinate changes within a second. These mickeys are representative of the packets of mouse data that are being generated within a second. As previously noted in Figure 3.2 and 3.3, the mickey value is needed for these formulas which calculate the mouse and pointer movement. Two tests were conducted, one with artificial acceleration disabled the other with it enabled.

The first test was conducted on the Windows 7 operating system with the artificial acceleration disabled in the mouse control pane as well as zeroed out values in the Windows registry.

**Figure 7.4 Sample Output with Artificial Acceleration Disabled**

Figure 7.4 depicts a small sampling of the report generated in a command prompt or terminal window. At the end of the report, a plot is generated of the session.
Figure 7.5 High Velocity Mouse Movement Model with Artificial Acceleration Disabled

The blue dots on the plot are coordinates that were captured during the recording. Given the speed of the mouse’s movement, it is possible to out run the amount of packets the mouse can generate which is represented by the spacing in between the points. The X and Y axis are dynamically sized based off of how far the mouse cursor travels out towards the limits of the monitor.

On the second test on the user report, the artificial acceleration thresholds are annotated so the proper functions can be manipulated during the analysis report.

Enter a user number: 12
Enter in your monitor size in inches: 24
Screen size: L*W: (1920, 1080)
MouseSpeed is 1
MouseThreshold1 is 6
MouseThreshold2 is 10

Figure 7.6 Artificial Acceleration Enabled user Prompt

These thresholds depict that during the simulation, if the velocity of the mouse passes the first threshold, the velocity of the pointer will increase by a factor of 2, if the velocity of the mouse passes the second threshold, the velocity of the pointer will increased by a factor of 4.

Figure 7.7 Sample Output with Artificial Acceleration Enabled

As depicted in the sample analysis report, it was a fairly common occurrence to pass the lower threshold during the simulation. Even trying to forcefully manipulate the mouse to reach the higher threshold proved to be fruitless as this would require a motion from the mouse that is not natural. With this observation and coupled with the fact that artificial acceleration is enabled by default, in future analysis, the lower threshold will be of the utmost importance as it is commonly surpassed.

Figure 7.8 Sample Output with Artificial Acceleration Enabled

Another observation that was apparent is the amount of distance between points in the plot depicted in Figure 7.8. There is a considerable more amount of spacing in between the points that is directly attributed to artificial acceleration. The bypassing of the lower threshold created
such vigorous acceleration that the programmatic means by which the recording of the coordinates was conducted is being overrun by the amount of data packets coming from the mouse’s sensors. Overall, these simulations created a more transparent view of artificial acceleration as the visual models show the drastic increase in speed and concurrently the flaws in the recording methods or the lack of bus speed from the mouse.

8. Suggestions for data collection

For more accurate results, some improvements must be made to the data collection methods. In order to properly calculate the mouse movements, several values must be retrieved from the operating system. For the Windows operating system, the screen resolution and the following values from the Windows Registry: MouseSpeed, MouseThreshold1, MouseThreshold2, must be queried. For the Mac OS X operating system, the screen resolution and the acceleration scaling value must be queried.

To get the appropriate results, two separate scripts were created. The first script (Figure 7.1) queries the operating system to determine the version and type of the operating system. If the returned information contains the phrase “Darwin” or “darwin” then the queries for the Mac OS X operating system are executed. Otherwise, the execfile function, built in to the Python programming language, will execute the second script (Figure 7.2), which performs the queries for the Windows operating system.

```python
import os
import platform
import sys
import time

root = Tkinter.Tk()

screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()

# os system("ls -l")
# os.system("defaults read com.apple.mouse.scaling "
print(platform.platform()), platform.system()
print sys.platform

if "Darwin" in sys.platform or "darwin" in sys.platform:
    print sys.platform
    print screen_width
    print screen_height
    print runmediavaluegetxos.system("defaults read com.apple.mouse.scaling "
    print mediavaluegetx
else:
    print sys.platform
    execfile('MouseSpeedOfWindows.py')
```

Figure 8.1 System Query Scripts 1

These scripts introduce new values that must be recorded into the csv file that contains the recorded information. My suggestion is that this be included into the first row for each new session of a user. This will provide useful information for the mouse movements as well as show of any changes in the screen resolution or acceleration between each session.

9. Conclusions & future research

Now that the underlying problem of artificial acceleration has become understood, research can be pursued on defining methods of which mouse movements can be actively used as a vital biometric, on par with retina scans and fingerprint recognition. On a more micro prospective, this understanding of artificial acceleration as well as the methods described in Section 7 will help to deduce the true mouse movements of the user.

As this research is still ongoing, more focus needs to be placed on polishing these methods and tailoring them for more practical means such as the mouse movement biometric system. The improvements on data collection as well as the artificial acceleration calculations will streamline the mouse movement biometric system as well as provide for more accurate results when used for user recognition. Lastly, migrating these changes to the web browser based method that Pace University currently uses will allow for a larger collection of data where in the future, machine-learning practices could be applied.

10. References thus far


