MIMIC: A Kinematic Theory-Based Synthesizer for Human-Like Mouse Movements

Alessandro Nicola Capriati

Abstract—This paper presents MIMIC, a sophisticated mouse movement synthesizer designed to emulate human hand movements based on the principles of the Kinematic Theory of the Human Hand. MIMIC is developed with the objective of challenging and adapting to modern anti-bot protections, which are increasingly prevalent and sophisticated in online environments. The core of MIMIC's functionality lies in its ability to accurately replicate the nuanced dynamics of human mouse interactions, making it a valuable tool for testing and improving the robustness of anti-bot systems.

Index Terms—bot, antibot, physiological models, kinematics, simulation, browser

1 Introduction

In the context of anti-bot detection systems, user interaction forms a critical factor in determining the authenticity of a session. A commonly employed strategy in these systems to calculate the validity of a user session involves analyzing mouse movements. These movements, characterized by their unique and unpredictable nature, provide a significant challenge for bots to reproduce, ensuring a robust line of defense against bot intrusion. However, the field of bot technology is continuously evolving, developing sophisticated methods to challenge these anti-bot systems. One such approach is the construction of biometric synthesizers, capable of generating human-like data. This chapter delves into the construction of the mouse movement synthesizer **MIMIC**, designed to mimic realistic user interactions and challenge anti-bot protections. In this paper, we will walk through the development process and considerations that led to its development. We will analyze the intricacies of synthesizing realistic mouse movement data and share the innovative techniques employed in the construction of MIMIC. We will also showcase the effectiveness of the synthesizer through a visual comparison between real and generated mouse movement data. The paper is organized as follows: The first section will delve into the neurophysiological basis of mouse movement, providing a foundational understanding of the science that informs our model. The second section will offer an overview of existing anti-bot defenses that focus on mouse movements, giving insight into the current landscape of bot detection. The third and fourth sections are dedicated to the construction of MIMIC, where we discuss the mathematical models, algorithmic considerations, and implementation specifics of our synthesizer. In the fifth section, we present the results of MIMIC's performance, visually comparing the synthesized data with real user mouse movements.

1.1 A NeuroPhysiological Explanation Of Human Mouse Movements

A Mouse Movement can be thought as a complex interplay between the Central Nervous System (CNS) and the Neuro-Muscular System (NMS) [28; 14]. At high level, when we are initially confronted with a certain task to perform, such as moving the mouse pointer to hit a target, the CNS receives inputs from the eyes regarding the distance, direction and width of that target and generates motor commands for the the NMS to execute. As task familiarity increases, hence acquiring experience on performing the same repetitive action, the CNS undergoes gradual improvements in generating the right motor controls[35; 4]. When a control input is executed, the sensorimotor system can sense the movement's outcome and compare this to the desired outcome[33; 6]. In the event of the latter being different from the actual outcome, the CNS is notified of the error, and generates corrective signals to adjust the original trajectory. This feedback mechanisms allows the CNS to learn from the error, thereby improving its ability to perform the task with greater accuracy.

In Fig. 1 it is reported a schematic control diagram of a mouse movement, where d is indicative of the noise.

This diagram is based on the Kinematic Theory of Human Hand Movement [18], which interprets a rapid human hand movement signal as an infinite number of linear subsystems linked in parallel, triggered by a Dirac-Impulse $U(t - t_0)$ at instant t_0 weighted by a distance parameter D_i, $i = 1, 2....$ The final signal entering the NMS actuator block is therefore the sum of the convolution of the linear subsystems. Under the right hypothesis, that of the subsystem being governed by a proportionality relationship, it has been shown [36; 44] that the NMS can be globally described as a linear system having a lognormal impulse response.

2 How Mouse Biometry Is Employed by Anti-Bot Defenses

To understand how mouse biometry is employed by anti-bot defenses at differentiating BOTs traffic from legitimate users, we have conducted a study on the Client-Side Javascript of popular Anti-Bot companies. Our analysis has led to several findings, including the techniques used by these companies to collect and analyze mouse movement data, the limitations of mouse movement biometry as detection systems, and the introduction of a novel synthesizer to generate human-like trajectories. However, it is important to acknowledge

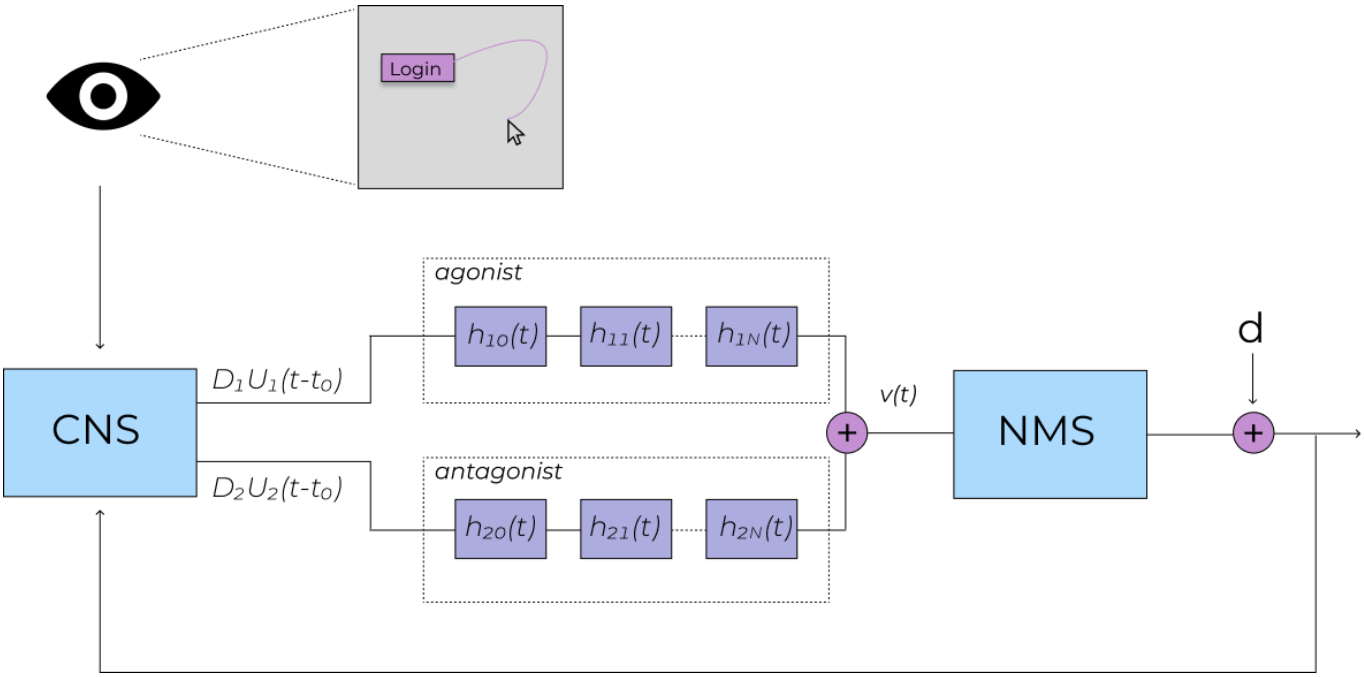


Fig. 1: Control diagram of a human mouse movement

the limitations of our approach. As we only had access to the obfuscated client-side code visible to users, we could only speculate on the potential workings of anti-bot systems based on our analysis of the code and previous literature.

2.1 Analysis

After careful analysis of the code and relevant literature, it has become apparent that anti-bot systems rely on a core set of metrics for their analysis. These metrics includes velocity, trajectory, duration, displacement, displacement angles, timestamps, mouse clicks. Out of these, velocity provides the most significant amount of unique data information [2]. These features are often used to train classifiers [3; 22]. Some other Anti-bots implementations adopt a combinations of these features to establish an overall threshold condition. Each metric contributes to this overall threshold, and once this threshold is reached, the system flags the user as a potential bot. For example, some implementations compute the ”straightness” of the curve and the ”number of segments” in a trajectory, and evaluate whether the samples of the trajectory in each segment follow a straight line. A ’segment’ refers to a portion of the mouse movement where the trajectory no longer follows the same linear pattern as before, that could start and end between two mouse clicks or when the direction or pattern of movement noticeably changes. Additionally, they may check the distance difference between two mouse click events, the sequence of click events, or the positions of the mouse clicks, increasing suspicious if the click happened at the edge of the screen, for instance, if the click occurs at as coordinates (0,0). Other implementations focus on the average velocity in a segment or the velocity of the entire stroke, making it more challenging for an attacker to bypass the system as it requires knowledge of the temporal information distribution of human mouse movements.

Another interesting approach is by leveraging digital identification techniques. By utilizing the robust XZERO or more recent IDELog[11] extraction techniques, it is possible to derive the optimal parameters that accurately capture the shape of the mouse movement trajectory and velocity. These extracted parameters can subsequently serve as inputs to the classifier to differentiate between well-composed and bad combinations of parameter values, thereby enhancing the effectiveness of anti-bot protections.

3 BUILDING MIMIC

MIMIC is designed as a controllable mouse mimicking solution based on the Sigma-Lognormal Model, which allows, with an high-degree of customization, to synthesize human-like mouse movements. Its core purpose is to serve as a proof of concept for testing the effectiveness of anti-bot protections. The Sigma-Lognormal model has been formally proven in literature to be reliable in generating trajectory almost indistinguishable from human-strokes, albeit it has been observed that the model applies a low-pass filter the generated trajectory, making it smoothed compared to human samples. As for digital signatures, this model has found many applications to generate forgery signatures [40], and with spectrum-analysis techniques[9]. In this section we introduce the process to develop MIMIC synthesizer, describing the solutions we choose to bypass various type of anti-bots protections.

3.1 Implementation

The architecture of MIMIC is depicted in Figure 2. MIMIC is designed to adapt to the unique characteristics of the device on which it is intended to be generated. To achieve this, the model requires device-specific settings, which are collectively referred to as **Configurations**. The Configurations includes:

- Frequency: the rate at which mouse samples are polled
- SNR: the signal-to-noise ratio
- Complexity: The number of strokes in the Sigma-Lognormal model
- Screen Dimension: the size of the device screen

3.2 Sigma Lognormal Model

The sigma-lognormal model describes the velocity profile resulting of a rapid human movement, as a vectorial summation of N lognormal curve[18; 20], where each lognormal is defined as:

$$\mathbf{v}(t; t_{0j}, \mu_j, \sigma_j^2) = \mathbf{D}_j \mathbf{\Lambda}(t; t_{0j}, \mu_j, \sigma_j^2) = \frac{1}{(t - t_{0j})\sigma_j\sqrt{2\pi}} e^{-\frac{(\ln(t-t_{0j})-\mu_j)^2}{2\sigma_j^2}} \tag{1}$$

$$\mathbf{s}(t) = \begin{bmatrix} x_r(t) \\ y_r(t) \end{bmatrix} = \sum \frac{D_j}{\theta_{ej} - \theta_{sj}} \begin{bmatrix} \sin(\phi_j(t)) - \sin(\theta_{sj}) \\ -\cos(\phi_j(t)) + \cos(\theta_{sj}) \end{bmatrix} \tag{2}$$

$$\phi_j(t) = \theta_{sj} + \frac{\theta_{ej} - \theta_{sj}}{2} \left(1 + \text{erf}\left(\frac{\log(t - t_{0j}) - u_j}{\sigma_j\sqrt{2}}\right)\right) \tag{3}$$

Equation 2 converts angles into circular arcs and overlaps them.

It is the foundation of our system model, and its parameters, $\sigma_j, \mu_j, D_j, t_j, t_{0j}$, will be regulated and stabilized by the **Controller**. The Controller generates the parameters in such a way that the synthesized movement meets specific criteria. In this section, we propose solutions to the following issues:

- Selecting parameters to link virtual target points together
- Constraining strokes within a specific screen size
- Constraining the velocity of a stroke

3.2.1 Linking Virtual Target Points

In this subsection, we introduce an innovative approach to control the starting and ending position of the Sigma Lognormal Model. The new equation obtained facilitates the linking of two consecutive virtual points, denoted as t_{p_j} and $t_{p_{j+1}}$, which acts as the starting and ending positions of a mouse stroke. By accurately simulating both the starting and ending positions of mouse clicks, we can mimic human-like interactions more convincingly. This capability is crucial when it is needed to interacts with HTML elements or executes clicks in element-free areas, as clicking on null elements might alert Anti-Bot systems. To address this issue, referring to Eq. 2 we solved the system

$$\begin{cases} x_f = \dfrac{D_j}{\theta_{ej} - \theta_{sj}} \left(\sin(\phi(t)) - \sin(\theta_{sj})\right) \\ y_f = \dfrac{D_j}{\theta_{ej} - \theta_{sj}} \left(-\cos(\phi(t)) + \cos(\theta_{sj})\right) \end{cases} \tag{4}$$

By solving Eq. 4 with respect to θ_{ej}, D we obtain Eq. 8 and Eq. 6.

By judiciously choosing values for the other parameters, we can determine appropriate values for θ_{ej} and D that allow the synthesis of a stroke ending at (x_f, y_f) for the j^{th} stroke.

3.2.2 Constraining strokes

All the anti-bot systems we examined incorporate a metric to verify if the submitted trajectory remains within the browser's window screen size where it originated. Hence, when generating synthetic mouse movements, it is essential to ensure that the terminal stroke position and clicks are within the boundaries of the screen size.

The problem can also be solved by generating parameters using Eq. 8 and Eq. 6. In this section we try to solve the same problem using a simulation approach. The problem can be described as follows:

$$\begin{cases} x_0 <= \dfrac{D_j}{\theta_{ej} - \theta_{sj}} \left(\sin(\phi(t_j)) - \sin(\theta_{sj})\right) <= x_f \\ y_0 <= \dfrac{D_j}{\theta_{ej} - \theta_{sj}} \left(-\cos(\phi(t_j)) + \cos(\theta_{sj})\right) <= y_f \end{cases} \tag{9}$$

where t_j represents the duration of the j-th stroke.

Given a randomly generated starting point (x_0, y_0), the goal is to identify suitable values for θ_{sj} and θ_{ej} such that the stroke's ending position (x_f, y_f) remains within the constraints of the device screen. To solve the system we propose a **simulated annealing** approach.

To define the neighbor function, we divide the device screen space into four quadrants, as illustrated in Figure 3b. Each quadrant is assigned a probability $p_j = A_j/A_{\text{total}}$, where A_j denotes the area of the j^{th} quadrant, and A_{total} represents the total device screen size. We partition the screen into 60 evenly spaced angles to generate two arrays, $\theta_{s_{\text{space}}}$ and $\theta_{e_{\text{space}}}$, representing the potential candidate values for θ_{sj} and θ_{ej}, respectively.

For the initial conditions, we randomly select θ_{sj} and θ_{ej} from their respective angle arrays. The neighbor function picks one of the quadrants using the corresponding probabilities p_j and applies a perturbation with a Gaussian noise function of μ 0 and σ of $\pi/6$ to the initial θ_{sj}. The nearest angle in the $\theta_{s_{\text{space}}}$ that matches the value is chosen as the new θ_{sj}, while five candidate angles θ_{ea} are drawn using a Monte Carlo simulation. An energy function is then implemented to evaluate the ending position for each angle combination.

The energy function calculates the differences between the ending position and each screen border. More precisely, we compute the differences between the ending position and the left, right, top, and bottom borders. As shown in Fig.3a, if the line drawn between the chosen position and the border is inside the rectangle, the difference results in a negative value and the energy is assigned a value of 0; if the difference is positive, then energy takes the value of the difference. All the energy values are then added together. The combinations of parameters that produce the neighbor with the lowest energy is chosen as the new configuration. This process repeats a maximum of 50 times, utilizing a Metropolis algorithm with an initial temperature of 1 to manage the annealing process. After each iteration, the θ_{sj} is removed from its $\theta_{s_{\text{space}}}$.

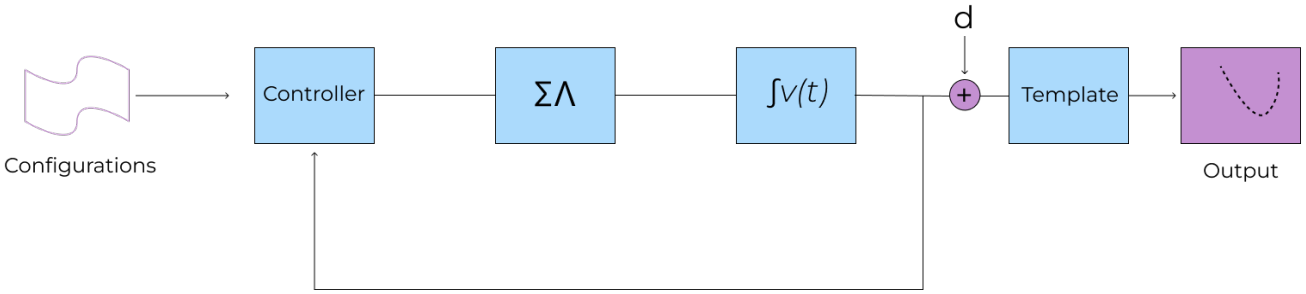


Fig. 2: Simplified Control diagram of MIMIC

The simulation halts when an angle combination gives, when evaluated using Eq. 2, an ending position that is confined within the screen size.

3.3 Frequency Generation

The term 'frequency' pertains to the rate at which the synthesized movement is sampled or, in the context of a browser, the rate at which mouse movement events are polled.

Our research concluded that the frequency at which a browser records mouse events is not consistent across various browsers, and may fluctuate over time depending on the browser's event-handling implementation and increases in CPU usage. While some browser implementations, such as Chrome 60+, have been observed to synchronize the user's polling rate with the display's refresh rate to enhance user experience, the frequency of mouse event registration can significantly vary across different browsers [43]. A comprehensive analysis of polling rate behavior across diverse browsers would necessitate substantial resources and remains a largely unexplored area. Consequently, in this study, we concentrate on analyzing the polling rate behavior on Chrome 112, using it as a reference to infer the behavior of other browsers.

For our experiment, we enlisted the participation of 20 individuals and directed them to connect to a site utilizing the tool mact-replay[8]. The participants were instructed to click on a random target on the screen. We observed that the mean frequency fluctuated within the range of [50, 140], with values oscillating in the interval of $[\mu - 3, \mu + 3]$.

Moreover, in specific stroke generations within the same session, we observed a sudden increase in the mean frequency of the movement. This could potentially indicate that the mouse events are polled at a faster rate during certain time intervals than others, which might be a sign of increased CPU usage. To account for this observation, during the implementation of the synthesizer, we actively manipulate the mean polling rate for each stroke. Specifically, we introduce random intervals where we increase or decrease the polling rate for a series of consecutive strokes. For instance, during the duration of a stroke, we define a mean μ and draw values from the interval $[\mu - 3, \mu + 3]$ using a Gaussian function with mean μ and sd 1,. In the subsequent stroke we set $\mu = \mu_1$, where μ_1 is a randomly generated value between $[50, 140]$, and repeat the process. This ensures that our synthetic data mimics the natural variation observed in human mouse movement data.

3.4 Constraining The Velocity

If we define the model with randomly generated parameters, there is a possibility that the output velocity could result in unrealistic human velocity profiles, which may perform too fast over a covered trajectory. Therefore, in this section we propose an algorithm which limits the upper bound of the velocity profile over a total time duration of t. We first calculate the global velocity maximum by taking the derivative of a Lognormal Function(Eq. 1) and set it equal to zero, obtaining:

$$t_{max} = t_0 + e^{-o^2+u} \tag{10}$$

$$v_{max} = \mathbf{D}j\mathbf{\Lambda}(t_{max}; t_{0j}, \mu_j, \sigma_j^2) \tag{11}$$

Where t_{max} is the time of the peak velocity, and v_{max} the peak velocity value.

$$\theta_{ej} = \frac{\theta_{sj}\,\mathrm{erf}\left(\frac{\sqrt{2}(u_j - \log(t - t_{0j}))}{2\sigma}\right) + \theta_{sj} + 4\,\mathrm{atan}\left(\frac{x_f \tan\left(\frac{\theta_{sj}}{2}\right) - y_f}{x_f + y_f \tan\left(\frac{\theta_{sj}}{2}\right)}\right)}{\mathrm{erf}\left(\frac{\sqrt{2}(u_j - \log(t - t_{0j}))}{2\sigma}\right) - 1} \tag{6}$$

$$D = \frac{x_f\left(\theta_{sj}\left(\mathrm{erf}\left(\frac{\sqrt{2}(u_j - \log(t - t_{0j}))}{2\sigma}\right) - 1\right) - \theta_{sj}\,\mathrm{erf}\left(\frac{\sqrt{2}(u_j - \log(t - t_{0j}))}{2\sigma}\right) - \theta_{sj} - 4\,\mathrm{atan}\left(\frac{x_f \tan\left(\frac{\theta_{sj}}{2}\right) - y_f}{x_f + y_f \tan\left(\frac{\theta_{sj}}{2}\right)}\right)\right)}{\left(\sin(\theta_{sj}) + \sin\left(2\,\mathrm{atan}\left(\frac{x_f \tan\left(\frac{\theta_{sj}}{2}\right) - y_f}{x_f + y_f \tan\left(\frac{\theta_{sj}}{2}\right)}\right)\right)\right)\left(\mathrm{erf}\left(\frac{\sqrt{2}(u_j - \log(t - t_{0j}))}{2\sigma}\right) - 1\right)} \tag{8}$$

We propose two strategies to restrict v_{max}. The first involves adjusting the value of μ, and the second involves adjusting the value of D.
If $v_{max} > v_{threshold}$, where $v_{threshold}$ is the maximum allowed velocity (default value set to 1500cm/s), a strategy is chosen at random. If $v_{max} < v_{thresholdmin}$, where $v_{thresholdmin}$ is set to 300cm/s, only the second strategy is utilized. In the case where $v_{thresholdmin} < v_{max} < v_{threshold}$, no adjustments are made.

Let's illustrate the first strategy, where we adjust the value μ to meet a specified value of v_{max}. We first solve Eq.1 for μ obtaining two solutions:

$$\begin{aligned}\mu_1 &= \lambda_1(\sigma, D, v, t, t_0) \\ &= -\sigma\sqrt{2\log\left(-\frac{D}{\sigma v(t_0 - t)}\right) - \log(2\pi)} + \log(-t_0 + t)\end{aligned}$$

$$\begin{aligned}\mu_2 &= \lambda_2(\sigma, D, v, t, t_0) \\ &= \sigma\sqrt{2\log\left(\frac{D}{\sigma v(-t_0 + t)}\right) - \log(2\pi)} + \log(-t_0 + t)\end{aligned}$$

Suppose we arbitrarily select the values of the parameters in Eq. 1, aiming to control the peak velocity. We begin by calculating the maximum velocity v_{max}. Should this maximum velocity exceed our predefined threshold, we then adjust our μ parameter. The adjustment involves choosing between μ_1 and μ_2, based on which value leads to a negative derivative of velocity at the time $t_{\max}$. This choice ensures that the velocity does not surpass our desired threshold.

The selection of a value for μ that yields a negative derivative is crucial for decreasing the maximum velocity of the function while avoiding the emergence of additional local maxima. For instance, consider the lognormal profile shown in Fig. 4a with parameters $\sigma = 0.83$, $\mu = -0.2$, $D = 8.5$, and $t_0 = 0$, that shows a maximum velocity of 7, which we aim to control and reduce to 6.5. If we choose a μ value that results in a positive derivative, as shown in Fig. 4c, it can lead to the unintended formation of new local maxima, evident in the appearance of a new peak around 0. This is not our desired outcome. In contrast, choosing a μ with a negative derivative, as we illustrate in Fig. 4b, would be the correct approach. This would successfully reduce the maximum velocity to our target value of 6.5, without introducing any new local maxima.

By repeating this process, we can create an interval of acceptable velocities corresponding to different thresholds. For example, to set a lower limit for the maximum velocity, we could apply the same method using a threshold of 3 instead of 6.5. This adjustment would lead to the determination of a new μ value, μ_{lower}, which results in a maximum velocity of 3. Combined with the earlier calculated μ_{upper}, which ensures that the maximum velocity does not exceed 6.5, we establish a range of μ values that effectively confines the maximum velocity within the desired interval of $[3, 6.5]$.

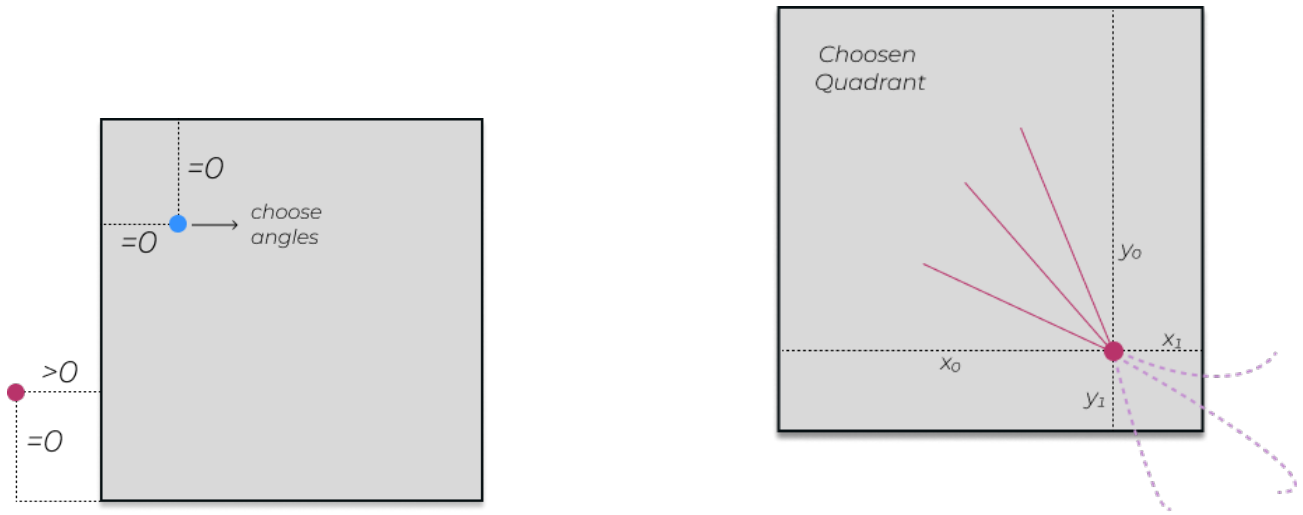
Finally, we intersect the range of μ values obtained for acceptable velocities with the feasible range of μ values, which, as demonstrated in Table 2, fall within the interval $[-3, 3]$, based on observations from real human data. This intersection results in a subset of μ values that are both realistic and suitable for generating an acceptable velocity profile for our synthetic mouse movements.

$$U = [\mu_{\text{lower}}, \mu_{\text{upper}}] \cap [-3, 3] \tag{12}$$

We obtain the interval of the feasible values of μ that satisfy the velocity constraints, where we sample values from for the lognormal curve.

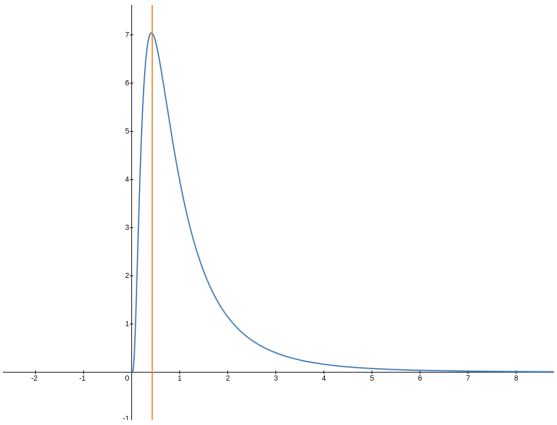
The **second strategy** for constraining the maximum velocity involves adjusting the parameter D. To enforce our velocity bounds, we redefine the limits for D as follows:

$$\begin{aligned} d_{min} &= \frac{v_{thresholdmin} \cdot D}{v_{max}} \\ d_{max} &= \frac{v_{threshold} \cdot D}{v_{max}} \end{aligned} \tag{13}$$

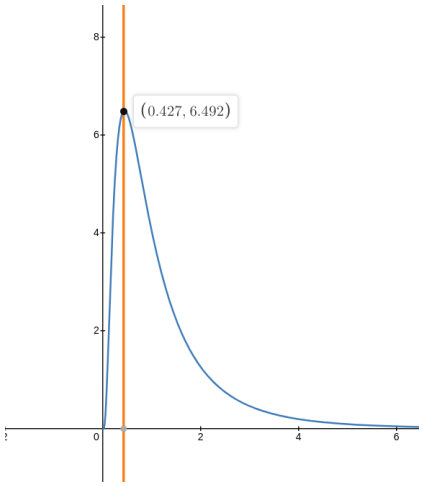


(a) (b)

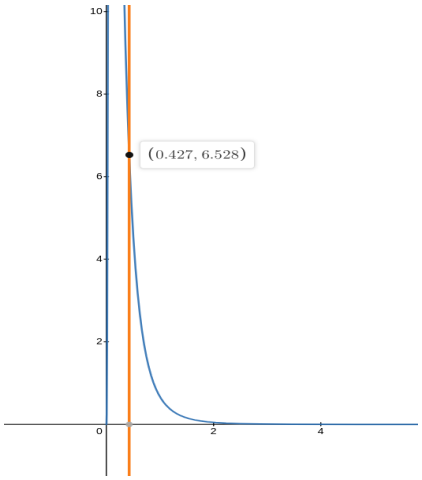
Fig. 3: Simulation process



(a) Initial Lognormal Profile



(b) Adjusted μ Lognormal with negative derivative



(c) Adjusted μ Lognormal with positive derivative

Fig. 4: Constraining Velocity Algorithm

The adjusted D values are then sampled from the intersection of the interval $[d_{min}, d_{max}]$ with the range $[100, 1200]$, as described in 2:

$$U = [d_{min}, d_{max}] \cap [100, 1200] \tag{14}$$

After implementing these strategies, we recompute $v_{\max}$. If the constraints on the maximum velocity are not satisfied, all parameters are regenerated and the process is repeated until a satisfactory set of parameters is achieved.

3.5 Noise Generation

As depicted in Fig. 2, Gaussian white noise (denoted by the symbol d) is added to the 2D trajectory upon its generation. The noise intensity is determined by the Signal-to-Noise Ratio (SNR) specified in the Configurations. To begin, the x and y components of the trajectory are separated, yielding two distinct signals. The power of each signal is calculated as S_x and S_y. We then compute the power of the noise, N_x and N_y, using the formula:

$$N_x = \frac{S_x}{SNR} \tag{15}$$

From this, we derive an expression for the Noise Amplitude, NA_x and NA_y:

$$NA_x = \sqrt{\frac{N_x}{\Delta T}} \tag{16}$$

Here, ΔT denotes the duration of a stroke.

Finally, we generate a Gaussian white noise process, $N(0, 1)$, and sample n values with n being number of polled samples of the stroke. This noise is then scaled by NA_x and NA_y and added to the respective x and y trajectory components. The result is a set of trajectories that have been polluted by white noise.

3.6 Parameters Tuning

This subsection elaborates on the generation process of the parameters: t, t_0, μ, σ, D, θ_{sj}, and θ_{ej}.

Table 1 displays the mean and standard deviation used to generate values for the parameters t and t_0 using a Gaussian random distribution. In this context, the parameter t represents the total duration of the motion, which is divided into N segments to accommodate the N lognormal curves.

Each lognormal adds an additional time duration to the total time, denoted as t_i, and has a starting time referred to as t_{0i}. The duration t_i for each curve is calculated using the Gaussian distribution, and the starting time t_{0i} marks the beginning of each lognormal curve's effect. For instance, the first lognormal curve starts at t_{00} and adds an additional time duration of t_0. If the sampled values for t, t_i, t_{0i} are lower than the minimum threshold, they are resampled.

Table 2 outlines the process for generating values via a uniform random distribution. Specifically, this table details the lower and upper bounds for each parameter. To ensure stability in our model, **additional constraints** are applied to the parameter θ_{ej}. It is restricted to a range that avoids values causing the results of Eq. 2 to become undefined. This range is defined as the intersection $\theta_{ej} \cap ([-\pi, \theta_{sj} - 0.2] \cup [\theta_{sj} + 0.2, \pi])$.

TABLE 1: Table of Parameters for Time

Parameter	Mean	Standard Deviation	Lower Bound
t (duration)	0.75	0.3	0.4
t_i (additional)	0.3	0.05	0.1
t_{0i}	$t_{0i-1} + \frac{t}{N}$	$\frac{t}{3N}$	0.1

TABLE 2: Table of Parameters

Parameter	Lower Bound	Upper Bound
u	-3	3
σ	0.1	3
D	100	1200
θ_{sj}	$-\pi$	π
θ_{ej}	$-\pi$	π

3.7 Mouse Events

Browsers register mouse movements through a series of events. These events are continuously polled, and each time an event is detected, its corresponding callback is executed. In our analysis, four key events relevant to anti-bot systems are taken into account [42]:

- **mousedown**: This event is triggered when a button on a pointing device is pressed while the pointer is inside the element.
- **mouseup**: This event is triggered when a button on a pointing device is released while the pointer is inside the element.
- **mouseclick**: This event is triggered when a button on a pointing device is clicked while the pointer is inside the element.
- **mousemove**: This event is triggered when a pointing device is moved.

Anti-bot systems also document the position where these events are triggered. If a click is initiated at a screen point where no trajectory exists, it suggests the possibility of a falsified mouse movement.

Additional verification measures include assessing click durations. As reported by [1], the typical mouse click duration is around 85ms on average, with upper and lower quartile values at 95ms (Q3) and 75ms (Q1), respectively. The upper and lower fences of this distribution are 135ms and 55ms. While anti-bot systems should not automatically flag as fraudulent mouse movements with click durations outside the interquartile range (IQR) as suspicious, a high frequency of such instances within a single session could indicate a potential attack.

Furthermore, the sequence of event registration in various browser implementations is an important factor. The expected order is: mousedown, mouseclick, mouseup. Any deviation from this sequence could be a strong indication of forgery.

The click locations are also critical. Typically, mouse clicks occur at points of local velocity minima, implying that users usually slow down before clicking a target. Therefore, we placed clicks on the trajectory at specific moments, denoted by t_{0j}. Since many anti-bot systems require at least two clicks on the trajectory for analysis, we randomly selected which instant t_{0j} to use for the click and delayed all other sample points by the duration of the click.

The duration of a click was calculated by sampling from a Gaussian distribution with a mean (μ) of 85ms and a standard deviation (σ) given by $\frac{IQR}{1.35} = 14.8148148$.

4 Browser Adaptation

4.1 Mouse Clicks

In our analysis of mouse profiles, we observed that the 'mousemove' events, which track the mouse's position, are triggered only when there is a change in the mouse's previous position. Additionally, each coordinate data point is rounded to the nearest integer. Therefore, if the mouse remains stationary, no 'mousemove' event is initiated.

We noticed that at low speeds, our synthesizer was generating many instances of repeated coordinates with different timestamps, a result of the browser's rounding of mouse coordinates to the nearest integer. This behavior is not unique to our synthesizer; it is also present in the browser environment. However, in typical browser usage, the frequency of repeated 'mousemove' events at the same position is generally limited to a maximum of three repetition. The probability of more than three repeated events at the same position diminishes progressively. Accordingly, we have adjusted our synthesizer's output to limit the number of repeating events to three, matching observed browser behavior.

Some anti-bot mechanisms further limit the collection of mouse coordinate samples to a maximum of n points and cease data collection after three mouse clicks. If these three clicks occur before reaching n samples, the anti-bot system stops collecting 'mousemove' events and sends the gathered sensor data to the server for analysis. Conversely, if the third click happens after accumulating more than n mouse samples, the system halts the collection of 'mousemove' events. In this case, only the coordinates and timestamps of the clicks are collected up to the third click

It is also necessary to discard all the clicks that are fired outside of the screen boundaries, as the browser cannot detect them.

4.2 Fullscreen Constraint

In Section 3.2.2, we described the methodology used to adapt the coordinates generated by the mouse synthesizer to a device screen of specific dimensions. The screen size

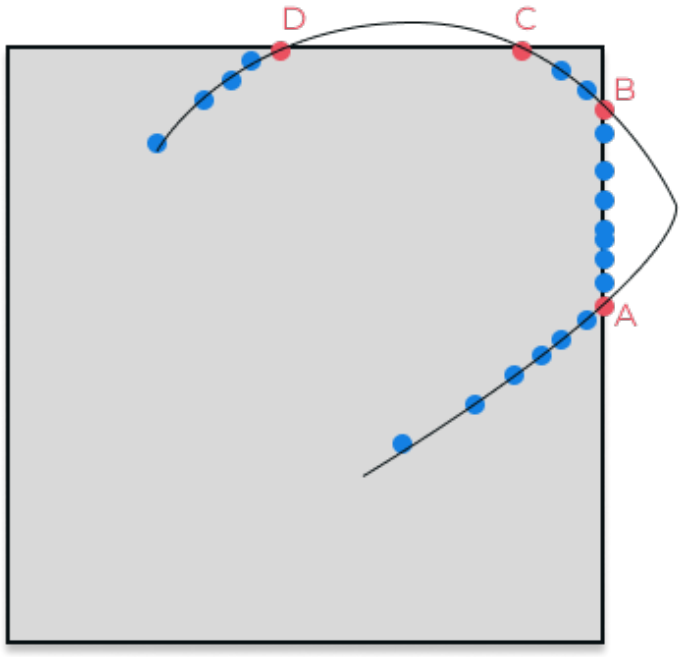


Fig. 5: Screen browser implementation

within a browser can be set in the Controller, allowing for either fullscreen or custom dimension settings.

In instances where the browser is in fullscreen mode, and the mouse samples exhibit an x-coordinate that does not fall within the range $0 < x <$ width, we adjust the x-coordinate to the respective screen boundary value. This adjustment is necessary because the screen width in fullscreen mode is equivalent to the monitor width of the user. As illustrated in Figure 5, if the mouse x-coordinate is 1500 and the screen width is 1000, the x-coordinate is adjusted to 1000. If the x-coordinate is less than 0, it is adjusted to 0. However, if the mouse y-coordinate does not adhere to the range $0 < y <$ height, we discard the corresponding samples. This approach is taken because the very top of the screen, typically occupied by the browser's navigation bar, is not a plausible space for mouse movement, and therefore no mouse movements are recorded in this area.

In scenarios where the browser is set to a custom screen size, any samples that exceed the boundaries in either the x or y dimensions are discarded, as illustrated in Figure 5 (segments C-D).

5 Results

In our investigation conducted in July 2023, we evaluated the effectiveness of both real human mouse trajectories and those synthesized using MIMIC against a popular high-security anti-bot solution, particularly focused on biometric protection events. Our metrics were the success rate and the total number of attempts. Each row of the table displays a different test, and the total attempt are the number of requests sent concurrently to the antibot.

We simulated human mouse movements using MIMIC, with the results presented in Table 4, using the simulation approach. These results show that MIMIC is capable of successfully emulating human mouse movements with a considerable degree of accuracy, achieving success rates ranging from 93% to 100%. Similar to the behavior observed with real human trajectories, we noted that an increase in the number of interactions with the website led to a slight decrease in the success rate. This trend suggests that the more requests made to the website, the more stringent the anti-bot system's scrutiny became, albeit the success rate remained above 90% for MIMIC.

This decline in success rate with increased requests could be due to various factors, such as certain nuances in browser implementation that were overlooked, or the anti-bot system's progressively refined analysis over time.

TABLE 3: Success rates and total attempts for real human trajectories

Success Rate (%)	Total Attempts
100.00	50
100.00	200
97.37	300
98.69	400
98.25	500
96.43	600
94.62	700
91.28	800
90.21	900
89.22	1000
88.81	1100
87.46	1200

TABLE 4: Success rates and total attempts with mimic

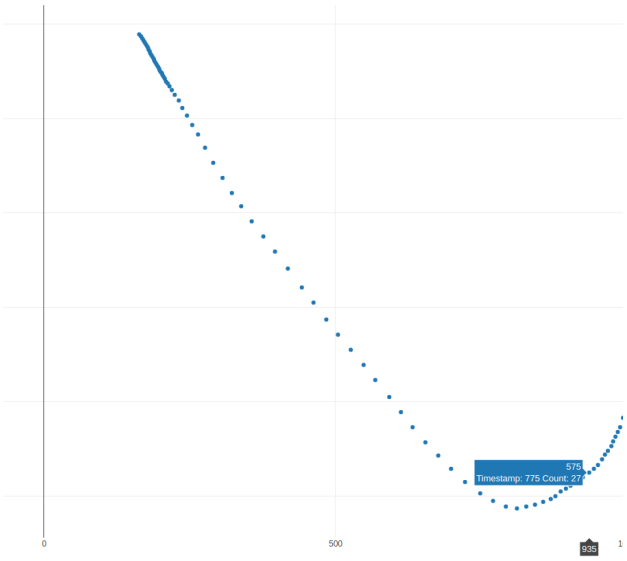
Success Rate (%)	Total Attempts
100.00	50
98.51	200
98.92	300
98.35	400
97.46	500
97.85	600
97.15	700
96.19	800
96.33	900
94.21	1000
93.32	1100
93.24	1200

We hypothesize that the slightly lower pass rate of the human mouse movements can be primarily attributed to the synthetic nature of the mouse clicks. Unlike natural user interactions, these clicks were generated artificially, which may have introduced subtle discrepancies detectable by the anti-bot systems.

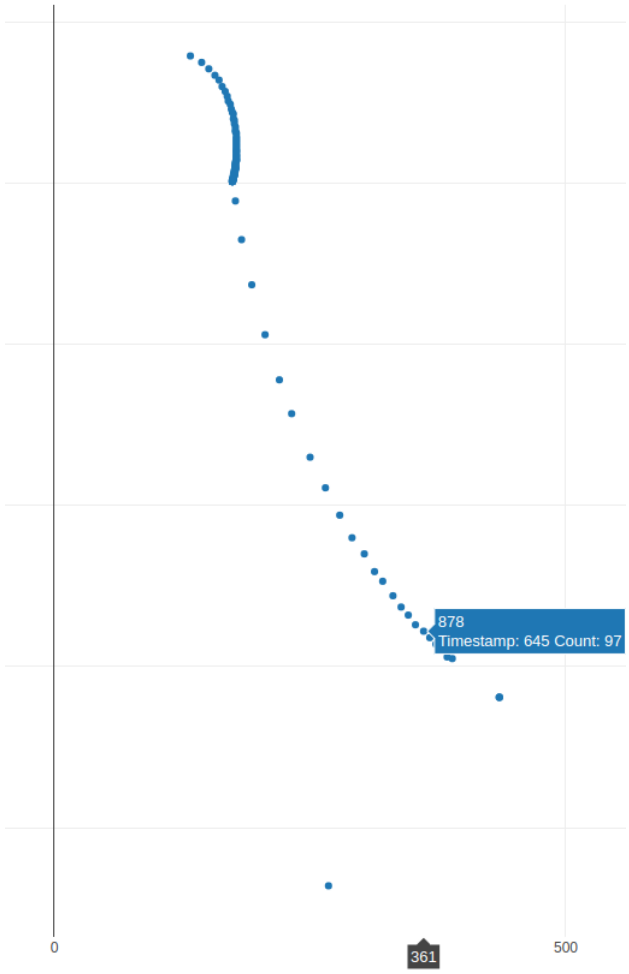
Additionally, Figures 6 offer a comparison between real mouse samples harvested from user interactions and synthetic mouse samples generated by MIMIC using the tool mact-replay. The plots for the synthetically generated mouse samples distinctshow the 'mouse clicks', not present in the real human movements.

6 Conclusion

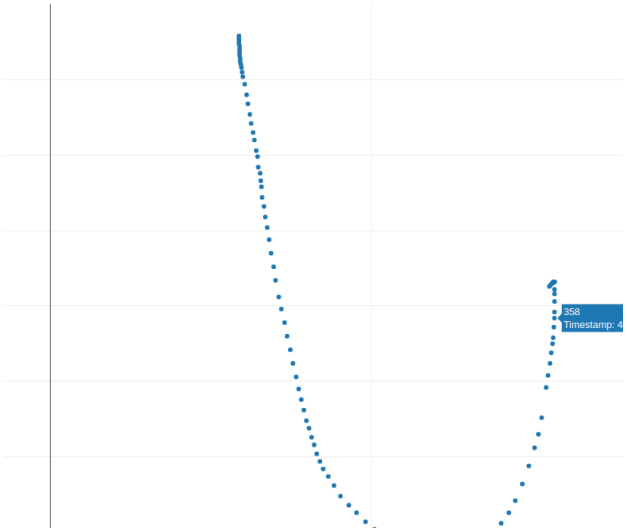
In conclusion, the evidence gathered from our study strongly supports the effectiveness of Mimic in replicating human-like mouse movements. The striking similarity between the patterns of real and synthetic mouse samples, as demonstrated in our analyses and visually represented

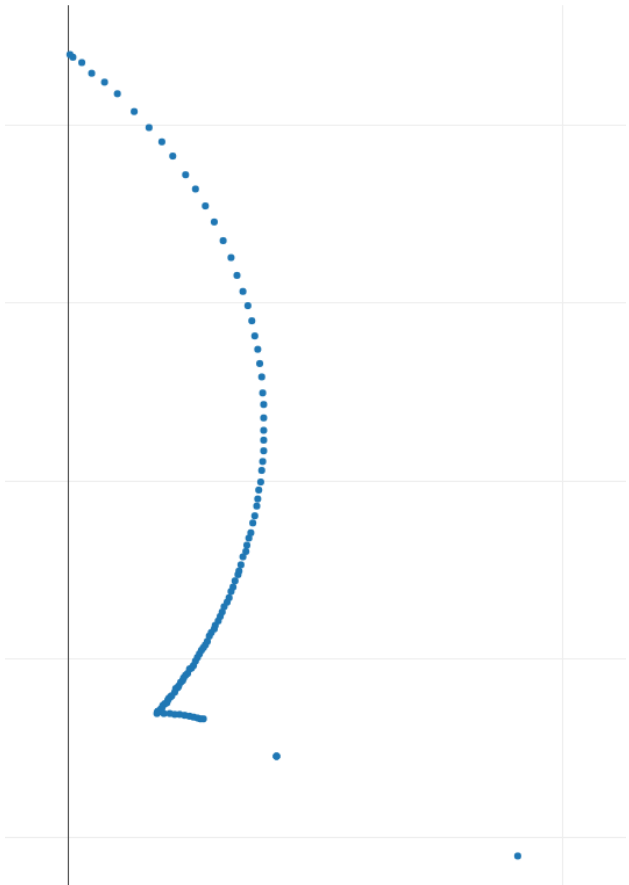


(a) Real Harvested Mouse Sample (Real)



(b) Generated MIMIC Mouse Sample (Mimic)





in the plots, underscores Mimic's capability in accurately mimicking these movements.

Moreover, the comparative data presented in our tables reveal that Mimic not only closely matches but, in some aspects, even surpasses the performance of real mouse movements. Despite these promising results, there remains room for improvement. One key area is the naturalness of the synthetic mouse clicks. Enhancing the realism of these clicks would be a crucial step towards achieving a near-perfect emulation of human mouse interaction. Additionally, we have observed instances where Mimic generates samples that are impractically close to each other, leading to less realistic trajectories. This issue points to a need for further refinement in the trajectory generation process. To address this, developing algorithms capable of identifying and rectifying less accurate trajectories could be a viable solution. Such algorithms would detect when the generated samples are too close to each other and either adjust these trajectories in real-time or regenerate them entirely to ensure a higher quality of simulation. Alternatively, investing more time and resources in the tuning of Mimic's parameters could also enhance the overall fidelity of the synthetic mouse movements.

These enhancements would not only improve the realism of the generated mouse movements but also contribute to making Mimic a more robust and reliable tool for simulating human mouse dynamics, solidifying its utility in environments where precise mouse movement emulation is critical. Additionally, the development of Mimic offers significant implications for anti-bot systems. By implementing Mimic, developers of anti-bot technologies can test the accuracy of their systems in detecting synthetic mouse movements. Furthermore, Mimic can be used to train and refine classifiers within these systems, thereby enhancing their ability to differentiate between human and non-human interactions

REFERENCES

[1] On mouse dynamics as a behavioral biometric for authentication, ASIACCS '11: Proceedings of the 6th ACM Symposium on Information, Computer and Communications SecurityMarch 2011Pages 476482, https://doi.org/10.1145/1966913.1966983

[2] Alejandro Acien, Aythami Morales, Julian Fierrez, and Ruben Vera-Rodriguez, "BeCAPTCHA-Mouse: Synthetic mouse trajectories and improved bot detection," *Expert Systems with Applications*, vol. 168, p. 114218, Elsevier, 2021.

[3] Zi Chu, Steven Gianvecchio, and Haining Wang, "Bot or Human? A Behavior-Based Online Bot Detection System," in *Proceedings of the 2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, Istanbul, Turkey, Aug. 2012, pp. 410-417. DOI: 10.1109/ASONAM.2012.86

[4] Bardella, W. A. and Enoka, R. M., "Human movement variability, nonlinear dynamics, and pathology: is there a connection?," *Hum. Mov. Sci.*, vol. 30, no. 5, pp. 869–888, Oct. 2011.

[5] Rosenbaum, David A., "Motor Control," in *Encyclopedia of Cognitive Science*, edited by L. Nadel, pp. 1–7, Nature Publishing Group, 2006.

[6] Soechting, John F. and Flanders, Mark, "Sensorimotor representations for pointing to targets in three-dimensional space," *J. Neurophysiol.*, vol. 62, no. 3, pp. 582–594, Sep. 1989.

[7] Aligned input events, Dave Tapuska https://developer.chrome.com/blog/aligning-input-events/

[8] Mact Reply, https://github.com/voidstar0/mact-replay

[9] Galbally, J. and Plamondon, R. and Fierrez, J. and Ortega-Garcia, J., "Synthetic on-line signature generation. Part I: Methodology and algorithms," *Pattern Recognition*, vol. 43, no. 6, pp. 2146-2158, 2010.

[10] Synthetic on-line signature generation. Part II: Experimental validation Javier Galbally, Julian Fierrez a, Javier Ortega-Garcia a, Re´jean Plamondon b

[11] Ferrer, Miguel A. and Diaz, Moises and Carmona-Duarte, Cristina and Plamondon, Rejean, "Iterative Dual Spatial and Kinematic Extraction of Sigma-Lognormal Parameters," *Proceedings of the 14th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2014.

[12] Lin, Jyh-Miin, "Python Non-Uniform Fast Fourier Transform (PyNUFFT): An Accelerated Non-Cartesian MRI Package on a Heterogeneous Platform (CPU/GPU)," *Journal of Imaging*, vol. 4, no. 3, 2018, pp. 51.

[13] Lin, Jyh-Miin and Chung, Hsu-Wen, "Pynufft: python non-uniform fast Fourier transform for MRI," *Building Bridges in Medical Sciences 2017*, St John's College, CB2 1TP Cambridge, UK, 2017.

[14] Kandel et al, *Principles of Neural Science*, McGraw-Hill, 2012.

[15] Campobasso, Michele and Allodi, Luca, "THREAT/crawl: a Trainable, Highly-Reusable, and Extensible Automated Method and Tool to Crawl Criminal Underground Forums," *Eindhoven University of Technology*, Eindhoven, The Netherlands, 2023.

[16] AppleVis Community, "AppleVis," 2023, *Online forum*, https://www.applevis.com/comment/147881comment-147881.

[17] "Muscle activation profiles based on the proportionality hypothesis of the Kinematic Theory of Human Movements," *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2020.

[18] Guerfali, W. and Plamondon, R., "A new method for the analysis of simple and complex planar rapid movements," *Motor Control*, vol. 4, no. 3, pp. 295-314, 2000.

[19] Slifkin, A.B. and Eder, J.R., "Trajectory evolution and changes in the structure of movement amplitude time series," *Exp Brain Res*, vol. 191, no. 3, pp. 293-306, 2008.

[20] Plamondon, R. and Feng, C. and Woch, A., "A kinematic theory of rapid human movements: Part IV: a formal mathematical proof and new insights," *Biological Cybernetics*, vol. 89, no. 2, pp. 126-138, 2003.

[21] Niu, H. and Chen, J. and Zhang, Z. and Cai, Z., "Mouse Dynamics Based Bot Detection Using Sequence Learning," *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1-8, 2017.

[22] Folch, S.E., "Web Bot Detection Using Mouse Movement," *Universitat Politècnica de Catalunya*, Master's thesis, 2016.
[23] Suchacka, G. and Cabri, A. and Rovetta, S. and Masulli, F., "Efficient on-the-fly Web bot detection," *Expert Systems with Applications*, vol. 130, pp. 61-73, 2019.
[24] Houk, J.C. and Rymer, W.Z., "Neural control of muscle length and tension," *Comprehensive Physiology*, 1981.
[25] Models for the Speed and Accuracy of Aimed Movements, David E. Meyer and J. E. Keith Smith
[26] Step-Tracking Movements of the Wrist in Humans. I. Kinematic Analysis Donna S. Hoffman and Peter L. Strick
[27] Minimum Jerk Trajectory Generation for Straight and Curved Movements: Mathematical Analysis Abdel-Nasser Sharkawy
[28] Latash, M.L., *Neurophysiological basis of movement*, Human Kinetics, 1998.
[29] Leiva, L.A. and Martín-Albo, D. and Plamondon, R., "Gestures à Go Go: Authoring Synthetic Human-like Stroke Gestures Using the Kinematic Theory of Rapid Movements," *Proceedings of the 2016 ACM on Interactive Surfaces and Spaces*, pp. 131-142, 2016.
[30] The Kinematic Theory Produces Human-like Stroke Gestures Luis A. Leiva, Daniel Martın-Albo Réjean Plamondon
[31] Relation Between EMG Activation Patterns and Kinematic Properties of Aimed Arm Movements, C. C. A. M. Gielen , K. van den Oosten F. Pull ter Gunne
[32] Trajectory evolution and changes in the structure of movement amplitude time series T Andrew B. Slifkin , Jeffrey R. Eder
[33] Diedrichsen, J., "Optimal Task-Dependent Changes of Bimanual Feedback Control and Adaptation," *Journal of Neurophysiology*, vol. 101, no. 4, pp. 1812-1825, 2009.
[34] Plamondon, R. and O'Reilly, C. and Rémi, C. and Duval, T., "The Lognormal Handwriter: learning, performing, and declining," *Frontiers in Psychology*, vol. 2, 2011.
[35] Wolpert, D. M. and Diedrichsen, J. and Flanagan, J. R., "Principles of sensorimotor learning," *Nat. Rev. Neurosci.*, vol. 12, no. 12, pp. 739-751, 2011.
[36] Plamondon, R. and Djioua, M., "The limit profile of a rapid movement velocity," *17th Biennial Conference on Motor Behavior*, 2008.
[37] Plamondon, R. and Djioua, M., "A multi-level representation paradigm for handwriting stroke generation," *Human Movement Science*, vol. 25, no. 4-5, pp. 586-607, 2006.
[38] Plamondon, R., "A kinematic theory of rapid human movements," *Biological Cybernetics*, vol. 72, no. 5, pp. 133-152, 1995.
[39] Gamboa, H. and Fred, A., "A Behavioural Biometric System Based on Human Computer Interaction," *Proceedings of the 5th WSEAS Int. Conf. on Applied Informatics and Communications*, pp. 20-22, 2004.
[40] Plamondon, R. and O'reilly, C. and Galbally, J. and Almaksour, A. and Anquetil, É., "Recent developments in the study of rapid human movements with the kinematic theory: Applications to handwriting and signature synthesis," *Signal Processing*, vol. 107, pp. 40-61, 2015.
[41] Marquez, William Khem, "ReverseJS," 2022, https://steakenthusiast.github.io.
[42] "Mouse events - Web APIs, MDN Web Docs," https://developer.mozilla.org/en-US/docs/Web/API/MouseEvent, [Accessed: 11-Jul-2023].
[43] Tapuska, Dave, "Aligned input events," 2022, https://developer.chrome.com/blog/aligning-input-events/.
[44] Plamondon, R. and Alimi, A.M. and Yergeau, P. and Leclerc, F., "Modelling velocity profiles of rapid movements: a comparative study," *Biol. Cybern.*, vol. 69, no. 2, pp. 119-128, 1993.