Emulating Human-Like Mouse Movement Using Bezier Curves and Behavioural Models for Advanced Web Automation

Sai Adarsh S[1], Prasanna Venkateshan[2], Prithvi Alva Suresh[3]
[1]*Graduate (2022), Department of Computing, Coimbatore Institute of Technology, Coimbatore-14.*
[2]*Graduate (2025), School of computing and Augmented Intelligence, Arizona State University, US-85281.*
[3]*Graduate (2021), School of Computer Science, Georgia Institute of Technology, Georgia, US-30332.*

***Abstract*—Advanced web automation systems such as browser agents address the challenge of bot detection by generating highly realistic mouse movements. This paper details the technical aspects of this emulation, which is fundamentally built upon cubic Bezier curves for path generation, complemented by velocity variations and movement timing derived from Fitts's Law. Furthermore, it explores the incorporation of stochastic micro-adjustments and contextual behaviours to enhance authenticity, providing a comprehensive understanding of how these systems produce trajectories and interaction delays that are indistinguishable from genuine human interactions, thereby countering common bot-detection mechanisms.**

***Index Terms*—Human-computer interaction (HCI), Bezier curves, Fitts's Law, mouse trajectory modelling, behavioural biometrics, web automation, motor control, speed–accuracy tradeoff.**

I. INTRODUCTION

Automation is increasingly used in recent web interactions, supporting tasks ranging from testing to data extraction. However, a significant challenge arises from the increasingly sophisticated bot detection systems designed to identify and block non-human actors. Traditional automation scripts often fall short, as they tend to trigger events programmatically or move cursors instantaneously, leaving clear digital footprints that distinguish them from genuine human users. These discernible patterns become a primary vector for bot detection.

To counteract this, recent web automation systems have developed sophisticated techniques to simulate continuous, realistic cursor paths. These systems utilise advanced mathematical models and behavioural algorithms to emulate human cursor dynamics with high fidelity, ensuring that automated interactions appear indistinguishable from those performed by a human. This paper delves into the technical mechanisms underpinning this advanced emulation, focusing on the mathematical models and behavioural algorithms that enable human-like mouse movement.

II. THE FUNDAMENTALS OF BEZIER CURVES

The cornerstone of realistic mouse trajectory generation in recent web automation systems is the cubic Bezier curve. Bezier curves are a well-established technique in computer graphics known for producing smooth, natural-looking paths. They enable the generation of a wide range of smooth trajectories between any two points, many of which closely resemble natural human movement.

THE MATHEMATICAL FORMULATION OF CUBIC BEZIER CURVES

For a path between a given start point P_0 and an end point P_3, a cubic Bezier curve (degree $n = 3$) is defined by the following mathematical formula:

$$\mathbf{B}(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t) t^2 P_2 + t^3 P_3$$

In this equation: $B(t)$ represents the calculated point on the curve at a given parameter t; $t \in [0,1]$ signifies the progression along the curve, where $t = 0$ corresponds to the start point P_0 and $t = 1$ corresponds to the end point P_3, P_0 is the initial start point of the cursor, P_3 is the target end point of the cursor, P_1 and P_2 are the crucial control points that dictate the curve's shape and curvature. These points

do not lie on the path itself but exert a pull on the curve, influencing its arc.

Figure 1: Illustration of a cubic Bezier curve with control points P_0 through P_3 used to generate a smooth trajectory between a start and end position.

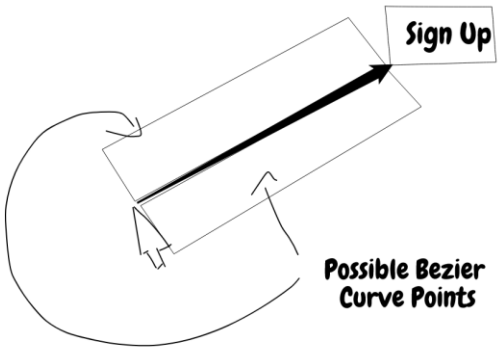

Figure 2: Control point placement for natural human-like curves. Points are randomly selected from one side of the cursor path to avoid unnatural 'S' shapes.

GENERATING NATURAL ARCS

To ensure that the generated cursor paths resemble natural human movement and avoid irregular or unnatural curves, the control points P_1 and P_2 are strategically randomised. When a system needs to move or click on a coordinate or element, it performs the following steps for path generation: (1) It generates two control points (P_1, P_2) with randomised offsets from the direct path between P_0 and P_3. (2) A critical technique for achieving natural arcs is that these randomised points are chosen from only one side of the line connecting P_0 and P_3. This constraint prevents abrupt or unnatural "S" shapes and ensures that the curve maintains a smooth, consistent arc, much like a human hand would naturally sweep. (3) Once the curve is defined by its four points, it is discretized into a specific number of steps, typically ranging from 50 to 100. For each step i, the next cursor position is precisely calculated by evaluating $B(i/n)$, where n is the total number of steps. Well calibrated implementations can even include timestamps for each point, generated based on a trapezoidal rule, to further enhance realism.

III. MIMICKING HUMAN DYNAMICS: VELOCITY AND TIMING

Beyond merely generating a curved path, realistic human-like mouse movement requires incorporating dynamics that reflect how humans accelerate, decelerate, and interact with targets.

VELOCITY VARIATIONS

Human mouse movements are rarely at a constant speed, they involve an initial burst of acceleration followed by deceleration as the target is approached. To mimic this, advanced systems apply velocity variations along the Bezier path. This is achieved using an acceleration formula such as:

$$v(t) = v_{\max} \cdot (1 - e^{-t/\tau})$$

This formula ensures non-linear speed, where v_{max} is the maximum velocity and τ is a time constant influencing the rate of acceleration and deceleration. By applying this model, the system ensures that the cursor does not move at a uniform pace, a behaviour commonly indicative of automation. Recent implementations also account for mouse speed, allowing for default random speeds or specific overrides via configurable options.

MOVEMENT TIMING WITH FITTS'S LAW

One of the most significant aspects of human-like interaction is the time taken to reach and interact with a target. This aspect is accurately modelled by Fitts's Law, a fundamental principle in human-computer interaction originally developed by Paul Fitts in 1954. Fitts's Law represents one of the most robust and widely validated models of human motor behaviour, demonstrating remarkable consistency across different limbs, input devices, physical environments, and user populations. The law is grounded in information theory, treating target acquisition as an information processing task. The fundamental insight is that movement difficulty can be quantified using the relationship between target distance and target size, creating what Fitts termed the *Index of Difficulty (ID)*. The basic formulation

quantifies the *movement time (MT)* required to rapidly move to a target area:

$$MT = a + b \cdot \log_2\left(1 + \frac{D}{W}\right)$$

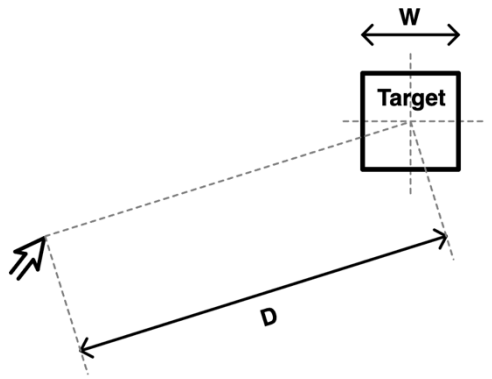

Figure 3: Illustration of Fitts's Law showing target width *W* and distance *D* used to compute movement time.

This equation embodies several critical insights about human motor control: *D* represents the distance from the current cursor position to the center of the target, capturing the spatial extent of the required movement, *W* represents the width or size of the target element measured along the axis of motion, functioning as the tolerance for error in the final position, *a* and *b* are empirically determined constants that reflect human motor processing and physical movement characteristics. The parameter a (typically ≈ 100ms) represents the intercept and is often interpreted as a baseline processing delay, while b (typically ≈ 150ms) represents the slope and describes the rate at which difficulty increases with task complexity. The logarithmic relationship captures a fundamental aspect of human motor control: the speed-accuracy tradeoff. As targets become smaller or more distant, users must slow down to maintain accuracy, but this relationship is logarithmic rather than linear. This means that doubling the difficulty does not double the movement time, reflecting the adaptive nature of human motor control systems. The law captures the two-phase nature of human pointing movements: an initial ballistic phase characterized by rapid but imprecise movement toward the target, followed by a corrective phase involving slower, more precise adjustments to acquire the target accurately. Advanced automation systems utilize this formula to calculate realistic movement durations. For example, to click a small button (*width W = 20px*) located at a distance *D = 500px*, the approximate movement time would be:

$$MT = 100 + 150 \cdot \log_2\left(1 + \frac{500}{20}\right) \approx 750\,\text{ms}$$

This granular control over timing is crucial because bots often execute movements and interactions far too quickly or with unnaturally consistent timing, making them easily detectable. Commonly used libraries similarly leverage Fitts's Law to determine the number of points to return in a path, relative to the target element's width and the distance between the mouse and the object, directly influencing the effective speed and duration of the movement.

IV. ADVANCED BEHAVIOURAL AUTHENTICITY

Beyond replicating the basic cursor path and general timing, human-like automation incorporates subtle imperfections and contextual behaviours characteristic of human interaction.

MICRO-ADJUSTMENTS AND JITTER

Human mouse movements are rarely perfectly smooth. They often include stochastic micro-adjustments, which are small, context-sensitive sub-movements akin to human jitter or spontaneous corrections. These subtle deviations introduce unpredictability and realism, making them difficult for automated scripts to reproduce and for detection systems to recognise as automated behaviour.

CONTEXTUAL BEHAVIOURS

Furthermore, well calibrated systems model realistic contextual behaviours that go beyond simple point-to-point movement: Hovering over elements: Instead of directly clicking, the cursor might pause over an element, mimicking a human user's brief consideration. Pausing near interactive UI components: Before engaging with dynamic elements like forms or menus, a human might pause for a moment. Gradual scrolling instead of abrupt jumps: The cursor's interaction with scrollable content is modelled to be smooth and continuous, avoiding the instantaneous viewport jumps typical of many

automated scripts. Realistic hover delays: Specifically for dropdowns or menus, systems insert realistic hover delays before a click, simulating human hesitation or review of options.

V. IMPLEMENTATION IN PLAYWRIGHT AND PUPPETEER

Prevailing browser automation frameworks like Playwright and Puppeteer allow low-level control of mouse events, which makes it possible to integrate custom movement logic based on Bezier curves. The goal is to simulate human-like cursor trajectories using code. In practice, this means generating a smooth, curved path between two points (rather than a straight line) and moving the mouse along that path in small increments, with added randomness for realism. By leveraging Bezier curve math for easing, a script can make the pointer accelerate and decelerate naturally during the move. This produces movement that mimics the way a real user might move their mouse, for example, starting slowly, speeding up in the middle of the motion, then slowing down as the cursor approaches the target. When augmented with subtle overshoots, jitter, and delays, the cursor ultimately follows a slow, curved, and slightly unpredictable trajectory akin to typical human behaviour.

IMPLEMENATION IN PLAYWRIGHT

Playwright's API provides a *page. mouse* object that can move the cursor to specified coordinates. While Playwright's default *mouse.move(x, y)* will instantly jump in a straight line, developers can write a custom routine to break a single move into many tiny steps, following a Bezier curve trajectory. For example, one can define a function to interpolate between the current mouse position and a target (x, y) using a cubic Bezier easing function. At each incremental step (perhaps dozens of steps for one full move), the script calculates an intermediate point along a curved path. A common technique is to use a Bezier easing curve like *bezier(0.25, 0.1, 0.25, 1)* (the control points for a smooth ease-in/ease-out) to get a progress fraction *t* that eases the motion. This eased progress ensures the cursor starts moving slowly, gains speed, then slows down as it reaches the destination, closely modelling natural hand movements.

Using Playwright's async capabilities, the script can await *page.mouse.move(x, y, {steps: 2})* for each tiny segment of the journey or simply call *mouse.move()* repeatedly in a loop for fine-grained control. After each small move, a short random delay is introduced (for instance, 10–60 milliseconds) to simulate the slight pauses and variations in human neuromotor response. Additionally, at each step a tiny random jitter can be added to the coordinates – for example, ±2–3 pixels off the ideal curve point. This means the cursor doesn't follow a mathematically perfect curve, but instead wobbles just a bit, introducing the same kind of imperfection a person would. Over the full motion, one might also incorporate a slight overshoot: if the target is far away, the script could intentionally aim a few pixels past the target and then move back, mimicking a user who slightly misses and corrects. By the end of the routine, the mouse is positioned exactly on the target and clicks. This approach, combining Bezier-based easing, random jitters, and brief pauses, makes the automated mouse movement appear organic. In fact, tests have shown that implementing Bezier-curved movements with jitter and randomised delays can make Playwright scripts appear significantly more human-like and help avoid detection.

IMPLEMENTATION IN PUPPETEER

In Puppeteer, achieving the same effect is equally feasible. Puppeteer also exposes a *page. mouse* that can be controlled in code. A developer could hand-code a similar Bezier interpolation loop in Puppeteer (the logic would be nearly identical to the Playwright approach described above, since both are JavaScript-based). Essentially, a sequence of intermediate *(x, y)* points is computed along a curved path, and the mouse is moved step by step using *page.mouse.move()*, incorporating random small offsets and delays at each step. However, the community has also created convenient libraries to handle this. One popular solution is Ghost Cursor, a Node.js library that wraps Puppeteer's mouse controls to automatically produce human-like movements. Ghost Cursor internally uses techniques such as Bezier curves and Fitts's law principles to generate realistic paths between coordinates. When *createCursor(page)* is used from Ghost Cursor, it attaches a custom cursor to the Puppeteer page that will move in a human-like way. For example,

invoking *cursor.move(selector)* does not teleport the pointer directly. Instead, the ghost cursor calculates a smooth curved route and traverses it with many tiny motions. It even intentionally overshoots or slightly misses the target if the travel distance is large, then corrects its position to land on the element. This overshooting behaviour is configurable via an overshoot threshold parameter, allowing an overshoot of up to a certain number of pixels beyond the target before readjustment. Similarly, Ghost Cursor randomises the exact landing position within the element's bounding box: rather than always clicking the precise center of a button or link, it selects a random point within the element (with an optional padding percentage to avoid edges) so that clicks occur at varying spots inside the target. The movement speed is also varied based on distance and target size, thereby avoiding a uniform motion speed. Under the hood, these features are powered by curved path generation and randomness to emulate the idiosyncrasies of real users.

To integrate such behaviour in a Puppeteer script, developers may either use Ghost Cursor (by simply replacing direct *page. mouse* calls with Ghost Cursor's *cursor.move()* and *cursor.click()* methods) or implement a custom Bezier movement function. Using the library is straightforward: after installation, a cursor is created with *const cursor = createCursor(page)* and then *await cursor.click(selector)* is invoked to move to the element and perform the click. Ghost Cursor handles the generation of the path and executes the intermediate movements and delays. If custom code is preferred instead, the pattern mirrors the Playwright approach, a Bezier curve path is calculated, and the script iterates through each point using *page.mouse.move()*, adding slight randomness to coordinates and timing. Regardless of the chosen approach, the outcome is that the cursor moves along a realistic trajectory, one that might gently arc toward the target, pause briefly, potentially overshoot by a small margin, and finally settle on the clickable element. These subtle movements make it significantly more difficult for anti-bot systems to distinguish the automation from a human user, as the mouse behaviour no longer appears robotic. By integrating Bezier curve-based movement logic into Playwright or Puppeteer scripts, developers can substantially enhance the realism of their automation, thereby increasing the likelihood of bypassing bot-detection heuristics that flag perfectly straight or excessively rapid mouse motions.

VI. IMPACT AND CONCLUSION

The sophisticated emulation of human-like mouse movement, as implemented by recent web automation systems, is paramount for ensuring compatibility with advanced bot detection systems. Prior research has consistently shown that automated agents exhibit distinguishable mouse dynamics, such as instantaneous movements, rigid trajectories, or consistent timing that deviate from human distributions. By meticulously generating trajectories using Bezier curves with randomised control points chosen from a single side, applying velocity variations based on acceleration formulas, and calculating interaction timing in accordance with Fitts's Law, these systems effectively neutralize a common and powerful vector for bot detection. The inclusion of stochastic micro-adjustments and a wide array of contextual behaviours, from overshooting and readjusting to randomising click targets within elements and incorporating realistic hover delays, further reinforces the authenticity of the simulated user.

In conclusion, the development of these advanced mouse emulation techniques represents a significant leap in web automation, allowing automated agents to seamlessly blend into the digital environment by producing trajectories, acceleration curves, and interaction delays that lie well within empirically observed human distributions. This capability is indispensable for operations requiring high fidelity interaction with web applications while minimizing the risk of detection and blocking.

REFERENCES

[1] P. M. Fitts, "The information capacity of the human motor system in controlling the amplitude of movement," Journal of Experimental Psychology, vol. 47, no. 6, pp. 381–391, 1954. DOI: 10.1037/h0055392

[2] I. S. MacKenzie, "Fitts' law as a research and design tool in human-computer interaction," Human-Computer Interaction, vol. 7, no. 1, pp.

91–139, 1992. DOI: 10.1207/s15327051hci0701_3

[3] E. R. F. W. Crossman and P. J. Goodeve, "Feedback control of hand movement and Fitts' law," Quarterly Journal of Experimental Psychology, vol. 35A, no. 2, pp. 251–278, 1983. DOI: 10.1080/14640748308402133

[4] S. N. Bernstein, "Démonstration du théorème de Weierstrass fondée sur le calcul des probabilités," Communications of the Mathematical Society of Kharkov, vol. 13, pp. 1–2, 1912.

[5] P. de Casteljau, "De Casteljau's autobiography: My time at Citroën," Computer Aided Geometric Design, vol. 16, no. 7, pp. 583–586, 1999. DOI: 10.1016/S0167-8396(99)00024-2

[6] G. Farin, Curves and Surfaces for Computer Aided Geometric Design, 4th ed., Academic Press Professional, 1997.

[7] T. D. DeRose, "Composing Bézier simplexes," ACM Transactions on Graphics, vol. 7, no. 3, pp. 198–221, 1988. DOI: 10.1145/44479.44482

[8] I. S. MacKenzie and W. Buxton, "Extending Fitts' law to two-dimensional tasks," in Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI '92), pp. 219–226, 1992. DOI: 10.1145/142750.142794

[9] N. Walker, D. E. Meyer, and J. B. Smelcer, "Spatial and temporal characteristics of rapid cursor-positioning movements with electromechanical mice in human-computer interaction," Human Factors, vol. 35, no. 3, pp. 431–458, 1993. DOI: 10.1177/001872089303500304

[10] R. W. Soukoreff and I. S. MacKenzie, "Towards a standard for pointing device evaluation: Perspectives on 27 years of Fitts' law research in HCI," International Journal of Human-Computer Studies, vol. 61, no. 6, pp. 751–789, 2004. DOI: 10.1016/j.ijhcs.2004.09.001

[11] J. Accot and S. Zhai, "Beyond Fitts' law: Models for trajectory-based HCI tasks," in Proc. CHI '97 Conf. Human Factors in Computing Systems, pp. 295–302, 1997. DOI: 10.1145/258549.258760

[12] J. Gori, O. Rioul, and Y. Guiard, "Reconciling Fitts' law with Shannon's Information Theory," ENS Cachan/Télécom ParisTech Conference Proceedings, France, 2015. HAL archive: hal-02300019

[13] I. S. MacKenzie, "A Note on the Validity of the Shannon Formulation for Fitts' Index of Difficulty," Open Journal of Applied Sciences, vol. 3, no. 6, pp. 360–368, 2013. DOI: 10.4236/ojapps.2013.36046

[14] J. D. Victor, "Approaches to Information-Theoretic Analysis of Neural Activity," Biological Theory, vol. 1, no. 3, pp. 302–316, 2006. DOI: 10.1162/biot.2006.1.3.302

[15] R. P. Heitz, "The speed-accuracy tradeoff: history, physiology, methodology, and behaviour," Frontiers in Neuroscience, vol. 8, p. 150, 2014. DOI: 10.3389/fnins.2014.00150

[16] R. P. Heitz and J. D. Schall, "Neural mechanisms of speed-accuracy tradeoff," Neuron, vol. 76, no. 3, pp. 616–628, 2012. DOI: 10.1016/j.neuron.2012.08.030

[17] D. E. Meyer, R. A. Abrams, S. Kornblum, C. E. Wright, and J. E. K. Smith, "Optimality in human motor performance: Ideal control of rapid aimed movements," Psychological Review, vol. 95, no. 3, pp. 340–370, 1988. DOI: 10.1037/0033-295X.95.3.340

[18] A. Nagamori, C. M. Laine, G. E. Loeb, and F. J. Valero-Cuevas, "Force variability is mostly not motor noise: Theoretical implications for motor control," PLoS Computational Biology, vol. 17, no. 3, p. e1008707, 2021. DOI: 10.1371/journal.pcbi.1008707

[19] A. M. van Mourik, A. Daffertshofer, and P. J. Beek, "Deterministic and stochastic features of rhythmic human movement," Biological Cybernetics, vol. 94, no. 4, pp. 233–244, 2006. DOI: 10.1007/s00422-005-0041-9

[20] C. J. Hasson, O. Gurelick, and G. Werhahn, "Neural Control Adaptation to Motor Noise Manipulation," Frontiers in Human Neuroscience, vol. 10, p. 59, 2016. DOI: 10.3389/fnhum.2016.00059

[21] G. M. Chaikin, "An algorithm for high-speed curve generation," Computer Graphics and Image Processing, vol. 3, no. 4, pp. 346–349, 1974. DOI: 10.1016/0146-664X (74)90028-8

[22] J. Peters, "Evaluation and approximate evaluation of the multivariate Bernstein-Bézier form on a regularly partitioned simplex," ACM Transactions on Mathematical Software, vol. 20, no. 4, pp. 460–480, 1994. DOI: 10.1145/198429.198434

[23] C. Yuksel, "Quadratic Approximation of Cubic Curves," Proc. ACM Comput. Graph. Interact. Tech., vol. 3, no. 2, Article 15, 2020. DOI: 10.1145/3406178

[24] E. Nava-Yazdani and K. Polthier, "De Casteljau's algorithm on manifolds," Computer Aided Geometric Design, vol. 30, no. 7, pp. 722–732, 2013.

[25] S. K. Card, W. K. English, and B. J. Burr, "Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a CRT," Ergonomics, vol. 21, no. 8, pp. 601–613, 1978. DOI: 10.1080/00140137808931762

[26] T. Grossman and R. Balakrishnan, "The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area," in Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI '05), pp. 281–290, 2005. DOI: 10.1145/1054972.1055012

[27] S. A. Douglas, A. E. Kirkpatrick, and I. S. MacKenzie, "Testing pointing device performance and user assessment with the ISO 9241, Part 9 standard," in Proc. CHI '99 Conf. Human Factors in Computing Systems, pp. 215–222, 1999. DOI: 10.1145/302979.303042

[28] R. Balakrishnan and I. S. MacKenzie, "Performance differences in the fingers, wrist, and forearm in computer input control," in Proc. CHI '97 Conf. Human Factors in Computing Systems, pp. 303–310, 1997. DOI: 10.1145/258549.258724

[29] M. Akamatsu and I. S. MacKenzie, "Movement characteristics using a mouse with tactile and force feedback," Int. J. Human-Computer Studies, vol. 45, no. 4, pp. 483–493, 1996. DOI: 10.1006/ijhc.1996.0063

[30] D. Elliott, S. Hansen, L. E. Grierson, J. Lyons, S. J. Bennett, and S. J. Hayes, "Goal-directed aiming: Two components but multiple processes," Psychological Bulletin, vol. 136, no. 6, pp. 1023–1044, 2010. DOI: 10.1037/a0020958

[31] J. E. Fischer, J. P. Cordeiro, and K. Howells, "Optimal Feedback Control for modeling human–computer interaction," ACM Trans. Comput.-Hum. Interact., vol. 29, no. 3, pp. 1–50, 2022. DOI: 10.1145/3524122

[32] L. A. Leiva and I. Arapakis, "The attentive cursor dataset," Frontiers in Human Neuroscience, vol. 14, p. 574675, 2020. DOI: 10.3389/fnhum.2020.574675

[33] P. J. Kieslich, F. Henninger, D. U. Wulff, J. M. Haslbeck, and M. Schulte-Mecklenbeck, "Mouse-tracking: A practical guide to implementation and analysis," in A Handbook of Process Tracing Methods, Routledge, pp. 111–130, 2019. DOI: 10.4324/9781315160559-9

[34] T. Yamauchi and K. Xiao, "Reading emotion from mouse cursor motions: Affective computing approach," Cognitive Science, vol. 42, no. 4, pp. 1333–1363, 2018. DOI: 10.1111/cogs.12557

[35] A. Binninger and O. Sorkine-Hornung, "Smooth Interpolating Curves with Local Control and Monotone Alternating Curvature," Computer Graphics Forum, vol. 41, no. 5, pp. 25–38, 2022. DOI: 10.1111/cgf.14600

[36] M. Hanik, E. Nava-Yazdani, and C. von Tycowicz, "De Casteljau's Algorithm in Geometric Data Analysis: Theory and Application," Computer Aided Geometric Design, vol. 118, Article 102220, 2024. DOI: 10.1016/j.cagd.2024.102220

[37] S. Vyas, M. D. Golub, D. Sussillo, and K. V. Shenoy, "Computation Through Neural Population Dynamics," Annual Review of Neuroscience, vol. 43, pp. 249–275, 2020. DOI: 10.1146/annurev-neuro-092619-094115

[38] R. Kopper, D. A. Bowman, M. G. Silva, and R. P. McMahan, "A human motor behaviour model for distal pointing tasks," Int. J. Human-Computer Studies, vol. 68, no. 10, pp. 603–615, 2010. DOI: 10.1016/j.ijhcs.2010.05.001

[39] D. U. Wulff, J. M. Haslbeck, P. J. Kieslich, F. Henninger, and M. Schulte-Mecklenbeck, "Mouse-tracking: Detecting types in movement trajectories," in A Handbook of Process Tracing Methods, Routledge, pp. 131–145, 2019. DOI: 10.4324/9781315160559-10

[40] B. E. Olivas-Padilla, S. Manitsaris, D. Menychtas, and A. Glushkova, "Stochastic-Biomechanic Modeling and Recognition of Human Movement Primitives, in Industry, Using Wearables," Sensors, vol. 21, no. 7, p. 2497, 2021. DOI: 10.3390/s21072497