Kinematic Features Are All You Need: Detecting Synthetic Mouse Trajectories Under Adversarial Optimization

Nick Tiwana
Voidware Studios
helzky@proton.me

Abstract

Synthetic mouse trajectory generators are a core component of aimbots and other automation tools that circumvent interactive security systems. Modern generators ground their output in motor-control science—Fitts' Law timing, lognormal submovements, physiological tremor—yet no prior work rigorously evaluates whether these biomechanical embellishments actually survive detection under adversarial pressure. We present a kinematic detection framework comprising 17 trajectory-shape features across four families (Fitts compliance, submovement morphology, kinematic smoothness, and geometry). On two public mouse dynamics datasets (29 users, 43,216 human trials from the Balabit Mouse Dynamics Challenge and the BOUN Mouse Dynamics Dataset), the framework achieves EER ≤ 0.001 and TPR $> 99.5\%$ at FPR $< 0.1\%$ under leave-user-out cross-validation against the strongest parametric generator we could construct. Even a logistic regression baseline achieves comparable discrimination, suggesting the human–synthetic separation is approximately linearly separable in this feature space. We evaluate adversarial robustness through a 5-round Bayesian optimization loop in which the attacker has white-box access to the feature set, unlimited queries, and a 17-dimensional parameter space. After a single round of detector retraining, the attacker's mean evasion score drops from 0.999 to 0.010 by round 5, with the slower convergence reflecting the greater distributional diversity of the expanded human population. Feature-family ablation reveals that this robustness arises from a tradeoff constraint: the parametric generator's single parameter set cannot simultaneously match all feature families, whereas the corresponding properties of human movement arise from distinct neuromuscular mechanisms. We additionally identify and exclude 15 confounded features arising from polling-rate artifacts in the Balabit dataset, establishing a cleaner evaluation methodology than prior work. Our results are limited to parametric generators evaluated on two mouse dynamics datasets; whether neural generation models (GANs, diffusion models) would produce similar outcomes remains open. Within these bounds, trajectory-shape detection appears to provide a practical defense layer for anti-cheat and interactive security systems.

1 Introduction

Synthetic mouse trajectory generation is widely regarded as a solved problem within the game cheating and automation communities. Modern generators produce smooth, Fitts-compliant pointing movements that pass visual inspection and defeat naive velocity-threshold detectors. In this paper, we examine whether this confidence is warranted.

The arms race between mouse automation tools and detection systems has intensified as interactive security measures increasingly rely on behavioral signals. CAPTCHA systems, anti-cheat engines, and fraud-detection pipelines all consume mouse input as evidence of human presence. Attackers have responded by building trajectory generators with progressively more physiological grounding: early approaches used simple Bézier interpolation; current state-of-the-art tools incorporate Fitts' Law timing, lognormal submovement decomposition, Ornstein–Uhlenbeck process noise, and speed-dependent tremor models [2, 1]. The implicit assumption is that if the generator's output "looks human" on the features the defender measures, detection becomes impossible.

We test this assumption empirically. We present a detection framework that, on the Balabit and BOUN mouse dynamics datasets (29 users combined), catches every class of parametric mouse generator we tested—including one we built specifically to evade detection—at TPR $> 99.5\%$ with FPR $< 0.1\%$. Under this evaluation, these generators are unable to reproduce the kinematic structure of real human movement, and adversarial parameter optimization does not close the gap.

Contributions. This paper makes four contributions:

1. A 17-feature kinematic detection framework that achieves EER ≤ 0.001 and TPR $> 99.5\%$ at FPR $< 0.1\%$ under leave-user-out cross-validation across two

independent datasets (29 users), using only trajectory-shape features—no timing statistics, no spectral analysis, no hardware fingerprinting. The detector operates on raw (x, y, t) data alone.

2. Evidence of adversarial robustness against parametric generators: a 5-round Bayesian optimization loop (60 Optuna trials per round across a 17-dimensional parameter space) fails to produce sustained evasion. After one round of detector retraining, the attacker's evasion score converges to 0.010 by round 5.
3. A taxonomy of 32 kinematic features across 6 families, with identification and exclusion of 15 confounded features arising from polling-rate artifacts, establishing a cleaner evaluation methodology for future work.
4. SigmaDrift, a motor-control-grounded parametric trajectory generator combining Fitts' Law timing, lognormal submovements, Ornstein–Uhlenbeck process noise, signal-dependent noise, and speed-dependent tremor. SigmaDrift serves primarily as an evaluation tool—the strongest parametric adversary we could construct—and is released as open-source infrastructure for future trajectory-detection research.

Scope and central finding. Our results apply to *parametric* trajectory generators—feed-forward pipelines that synthesize movement from fixed parameter distributions. Within this class, which includes all publicly known generation approaches from Bézier interpolation through motor-control-inspired models, our detector achieves near-perfect discrimination that resists adversarial optimization. We do not claim this extends to all possible generation architectures; in particular, neural generators trained directly on human data represent an untested threat that could narrow or close the gap (Section 8). Our contribution is establishing that the current generation paradigm is insufficient and characterizing *why* it fails at the feature level.

The remainder of this paper is organized as follows. Section 2 surveys related work on trajectory generation and detection. Section 3 defines the threat model. Section 4 describes SigmaDrift. Section 5 presents our feature taxonomy and confound analysis. Section 6 details the detection framework and main results. Section 7 evaluates adversarial robustness. Section 8 discusses implications, limitations, and future work. Section 9 concludes.

2 Background & Related Work

2.1 Mouse Trajectory Generation in Cheats

Mouse trajectory generation has evolved through several architectural paradigms, each adding physiological fidelity in response to improved detection.

Wind/gravity models. The WindMouse algorithm, originating in the RuneScape botting community circa 2010, combines Bézier-like interpolation with stochastic "wind" and deterministic "gravity" vectors [17]. At each simulation step, a wind vector introduces random lateral deviation while a gravity vector attracts the cursor toward the target. The approach produces visually plausible curves but has no physiological basis: it does not respect Fitts' Law, produces the wrong speed profile shape (monotonically accelerating rather than bell-shaped), and lacks any submovement structure. We include WindMouse as a baseline to demonstrate that even naive generators have distinct failure modes from physiologically-grounded ones.

Bézier and spline interpolation. Commercial aimbots frequently use cubic Bézier or B-spline paths with randomized control points. These produce smooth trajectories with controllable curvature but lack submovement decomposition, correction dynamics, and realistic timing. The speed profile along a Bézier curve is determined entirely by control point placement and parameterization speed, bearing no relation to the acceleration–deceleration patterns of aimed human movements.

Motor-control-inspired generators. The most sophisticated generators draw on the motor control literature. Plamondon's kinematic theory [2] models handwriting and pointing movements as superpositions of lognormal velocity profiles, each corresponding to an agonist–antagonist muscle pair activation. Flash and Hogan's minimum-jerk model [3] predicts the smooth, bell-shaped velocity profile observed in unperturbed reaching movements. SigmaDrift combines these ideas: it uses Fitts' Law for overall timing, lognormal submovements for the velocity profile, Ornstein–Uhlenbeck process noise for lateral deviation, signal-dependent noise following Harris and Wolpert [4], and speed-dependent tremor. It represents the most complete open-loop parametric generator we are aware of.

Gap. No prior work systematically evaluates whether physiologically-grounded generators actually evade kinematic detection, or whether adversarial optimization over generator parameters can close the gap.

2.2 Mouse Dynamics for Bot Detection

Behavioral biometrics. Mouse dynamics have been extensively studied for user authentication [9, 10]. These systems build per-user models from features such as movement speed, click patterns, and trajectory curvature. The goal is identity verification—distinguishing Alice from Bob—rather than the human-vs-bot classification we consider. While there is feature overlap, the evaluation method-

ology differs fundamentally: authentication systems measure equal error rates across users, whereas we require near-zero false positive rates at the population level.

Bot detection. Acien et al. introduced the BeCAPTCHA-Mouse challenge [11], training classifiers to distinguish human mouse trajectories from synthetic ones. However, their synthetic data was generated by simple models (Bézier interpolation, basic noise injection) that would not challenge a detector using kinematic features. More importantly, their evaluation did not assess adversarial robustness—whether an attacker who knows the features can tune a generator to evade.

Anti-cheat systems. Production anti-cheat engines (Easy Anti-Cheat, Vanguard, BattlEye) operate at kernel level with access to driver telemetry, hardware fingerprints, memory scans, and process integrity checks. Public knowledge of their mouse-trajectory analysis is limited to community reverse engineering. No rigorous academic evaluation examines which trajectory features survive adversarial pressure from strong generators. Our work fills this gap by evaluating detection under a white-box threat model with the strongest parametric adversary we could build.

2.3 Kinematic Features of Human Movement

Fitts' Law. The foundational relationship in aimed movement, Fitts' Law [1] predicts that movement time MT scales logarithmically with the index of difficulty: $\text{MT} = a + b \cdot \log_2(D/W + 1)$, where D is target distance and W is target width. This relationship holds across effectors (arm, wrist, finger), input devices, and populations, with typical $R^2 > 0.95$ in controlled experiments.

Submovement theory. Meyer et al.'s optimized submovement model [5] and Plamondon's kinematic theory [2] both predict that aimed movements consist of a primary ballistic phase followed by one or more corrective submovements. The number, timing, and amplitude of corrections depend on task difficulty and individual motor noise characteristics. These corrections produce multi-peaked velocity profiles that are a hallmark of human pointing.

Smoothness metrics. Kinematic smoothness—the absence of abrupt changes in the movement trajectory—is quantified by metrics such as root-mean-square jerk, normalized jerk, and the log dimensionless jerk (LDLJ) [6, 8]. Originally developed for motor rehabilitation assessment, these metrics capture the "quality" of movement coordination. The SPARC metric [7] provides a spectral smoothness measure, though we do not use it due to polling-rate confounds discussed in Section 5.2.

Key insight. The features we leverage capture *structural* properties of motor control—correction loop dynamics, neural noise characteristics, jerk profiles—that emerge from a closed-loop neuromuscular system processing visual feedback in real time. Our hypothesis, tested empirically in this paper, is that a parametric model with a fixed feed-forward architecture has difficulty replicating these properties through parameter tuning alone, because the relevant structure is an emergent property of the control architecture rather than a distributional parameter.

3 Threat Model

We evaluate under a strong, realistic attacker to establish an upper bound on parametric evasion capability.

Attacker. The attacker has (1) full knowledge of the 17-feature detection pipeline (open-source code), (2) the ability to build and tune arbitrary parametric generators, (3) access to public human trajectory datasets for calibration, and (4) computational budget for Bayesian optimization over generator parameters with unlimited detector queries. The attacker must inject events through the OS input stack, producing (x, y, t) triplets. The attacker cannot replay recorded human trajectories (orthogonal to our scope), modify the detection system (we assume standard anti-cheat integrity), or alter the observation channel.

Attacker goal. Generate synthetic aimed trajectories classified as human at the *per-trajectory* level—strictly harder than session-level evasion.

Defender. The defender observes raw (x, y, t) mouse events, segments movements at rest periods [9], and trains on labeled human data. The detector requires no hardware fingerprinting, driver-level telemetry, or timing side channels.

Scope. This threat model covers parametric generators with white-box feature access. It does not cover neural generators (GANs, diffusion models), which represent a distinct and potentially stronger threat class (Section 8).

4 The SigmaDrift Generator

SigmaDrift serves two roles in this paper: it is the adversary against which we evaluate our detection framework,

and it is released as open-source infrastructure for future trajectory-detection research. We designed it to be the strongest parametric generator we could build, grounded in motor control science, so that our detection results represent an upper bound on what current parametric approaches can achieve. (The features used to evaluate SigmaDrift are detailed in Section 5; readers primarily interested in the detection methodology may prefer to read that section first.)

4.1 Architecture Overview

SigmaDrift implements a six-stage feed-forward pipeline:

1. **Fitts' Law timing:** Compute movement time from target distance and width.
2. **Lognormal submovement decomposition:** Generate primary and corrective velocity profiles.
3. **Ornstein–Uhlenbeck noise:** Add mean-reverting lateral deviation.
4. **Speed-dependent tremor:** Inject 8–12 Hz sinusoidal component modulated by instantaneous speed.
5. **Signal-dependent noise:** Add velocity-proportional Gaussian noise.
6. **Endpoint correction:** Model undershoot/overshoot with corrective submovements.

Each stage has a physiological justification rooted in the motor control literature. Figure 1 shows example trajectories and velocity profiles from SigmaDrift alongside human data and WindMouse output.

4.2 Component Descriptions

Fitts' Law timing. Movement time is sampled as $\mathrm{MT} = (a + b \cdot \mathrm{ID}) \cdot e^{\mathcal{N}(0,0.08)}$, where $\mathrm{ID} = \log_2(D/W + 1)$ is the Shannon formulation of the index of difficulty. The multiplicative noise produces realistic trial-to-trial variability while preserving the Fitts relationship in expectation. We calibrate a and b from the Balabit dataset, yielding $a = 200$ ms and $b = 260$ ms/bit.

Lognormal submovements. Following Plamondon's kinematic theory [2], the velocity profile is composed of a primary lognormal pulse plus zero to two corrective submovements. Each submovement i has the form:

$$v_i(t) = \frac{D_i}{\sigma_i \sqrt{2\pi}(t - t_{0,i})} \exp\left(-\frac{(\ln(t - t_{0,i}) - \mu_i)^2}{2\sigma_i^2}\right) \quad (1)$$

where D_i is the distance covered, μ_i and σ_i control the timing and shape, and $t_{0,i}$ is the onset time. The primary submovement intentionally undershoots (92–97% of distance) or overshoots (15% probability, 102–108% of distance) the target, triggering one or two corrective submovements.

Ornstein–Uhlenbeck process noise. Lateral deviation is modeled as an OU process with reversion rate θ and volatility σ_{OU}:

$$d\eta(t) = -\theta \cdot \eta(t)\, dt + \sigma_{\mathrm{OU}}\, dW(t) \quad (2)$$

This produces mean-reverting perturbations that model the stochastic component of neural motor commands. The OU process ensures that deviations are bounded (unlike a random walk) while maintaining temporal correlation (unlike i.i.d. noise).

Speed-dependent tremor. Physiological tremor at 8–12 Hz is injected as a sinusoidal component with amplitude modulated by a sigmoidal speed gate: $A_{\mathrm{trem}}(t) = A_0/(1 + 0.3 \cdot v(t))$. This models proprioceptive suppression—the observation that tremor is attenuated during fast movements when the limb's inertia mechanically dampens oscillations [15].

Signal-dependent noise. Following Harris and Wolpert's seminal observation that motor noise scales with signal amplitude [4], we add Gaussian noise proportional to instantaneous velocity: $\varepsilon_{\mathrm{SDN}}(t) \sim \mathcal{N}(0, k_{\mathrm{SDN}} \cdot v(t))$. The parameter k_{SDN} controls the noise gain.

Curvature perturbation. A smooth lateral offset via a curvature-scale parameter prevents dead-straight paths. The offset follows a $s^2(1-s)^3$ profile (where s is normalized progress), producing maximum curvature during the acceleration phase, as observed in human pointing [16]. Vertical movements receive larger curvature due to wrist/forearm geometry.

4.3 Parameter Space

Table 1 lists all 17 tunable parameters with their default values and search ranges used during adversarial retuning (Section 7). Parameters are organized by the pipeline stage they control.

4.4 Comparison to WindMouse

WindMouse serves as a lower-bound baseline—a generator with no physiological grounding. Figure 2 illustrates the difference: WindMouse trajectories violate Fitts' Law by approximately 3× (movement times are far too fast for a given index of difficulty), produce the wrong speed-profile shape, and lack correction dynamics entirely. Notably, WindMouse and SigmaDrift are detected for *different reasons*: WindMouse fails on Fitts compliance and timing,

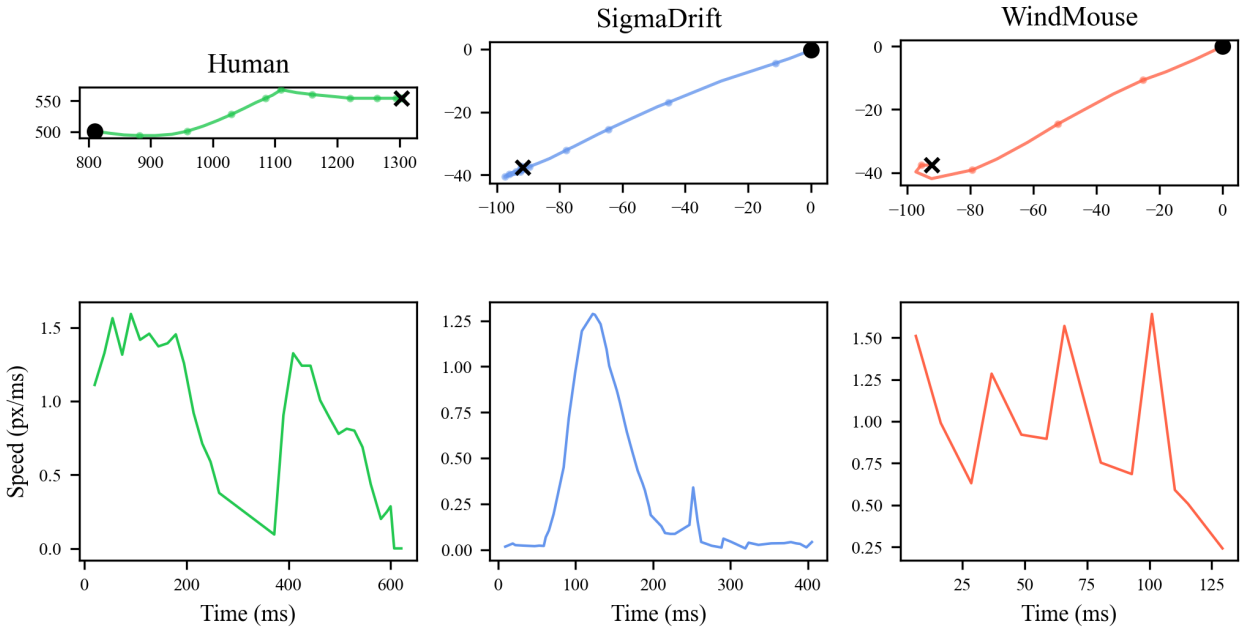

Figure 1: Example trajectories (top) and speed profiles (bottom) for human, SigmaDrift, and WindMouse movements. The human trajectory exhibits multiple velocity peaks with corrective submovements, irregular curvature, and high path variability. SigmaDrift produces a smoother, more stereotyped velocity profile with one dominant peak and small synthetic corrections. WindMouse shows an erratic spatial path with a non-physiological speed profile (multiple acceleration bursts without the characteristic bell shape). Start positions are marked with $\times$, endpoints with $\bullet$.

while SigmaDrift fails on smoothness and correction structure. This observation is important for understanding the generality of our feature set.

5 Feature Taxonomy

We initially extracted 32 features across 6 families from each trajectory. After careful confound analysis, 15 features were excluded, leaving a clean set of 17 features that measure trajectory shape and kinematics without relying on sampling-pipeline artifacts.

5.1 Feature Families

Fitts features (4). Movement time (`fitts_mt`), index of difficulty (`fitts_id`), Fitts residual (`fitts_residual`), and residual ratio (`fitts_residual_ratio`). These capture the speed–accuracy tradeoff: whether the trajectory's duration matches what Fitts' Law predicts for its distance and target width. The residual features are particularly informative, as they measure deviation from the population-level Fitts relationship, capturing individual trial-level variability that generators must reproduce.

Submovement features (5). Number of speed peaks (`sub_peak_count`), primary submovement amplitude ratio (`sub_primary_amp_ratio`), correction onset time (`sub_correction_onset`), inter-peak interval coefficient of variation (`sub_interpeak_cv`), and secondary/primary peak speed ratio (`sub_peak_speed_ratio`). These capture the correction dynamics that are central to human aimed movement: how many corrective submovements occur, when they begin, and how large they are relative to the primary movement.

Smoothness features (4). Root-mean-square jerk (`smooth_jerk_rms`), normalized jerk (`smooth_norm_jerk`), log dimensionless jerk (`smooth_ldlj`) [6], and curvature change rate (`smooth_curvature_change_rate`). These quantify the "quality" of movement coordination. Human movements are smooth within submovements but exhibit characteristic jerk at transitions between primary and corrective phases. Parametric generators tend to produce either too-smooth trajectories (missing the correction transients) or too-jerky ones (from additive noise models).

Geometry features (4). Path efficiency (`geo_path_efficiency`), maximum perpendicular

Table 1: Generator parameter space. Default values are calibrated to the combined human population. Ranges define the search space for adversarial optimization.

Parameter	Default	Range
Submovement shape		
`peak_time_ratio`	0.35	[0.20, 0.50]
`primary_sigma_min`	0.18	[0.10, 0.40]
`primary_sigma_max`	0.28	[0.15, 0.50]
`correction_sigma_min`	0.12	[0.08, 0.40]
`correction_sigma_max`	0.20	[0.10, 0.50]
Endpoint control		
`undershoot_min`	0.92	[0.80, 0.98]
`undershoot_max`	0.97	[0.90, 1.00]
`overshoot_prob`	0.15	[0.05, 0.60]
`second_corr_prob`	0.25	[0.10, 0.60]
Noise model		
`ou_theta`	3.5	[0.5, 10.0]
`ou_sigma`	1.2	[0.3, 6.0]
`sdn_k`	0.04	[0.01, 0.20]
Tremor		
`tremor_freq_min`	8.0	[6.0, 10.0]
`tremor_freq_max`	12.0	[10.0, 14.0]
`tremor_amp_min`	0.15	[0.05, 0.50]
`tremor_amp_max`	0.55	[0.10, 1.00]
Geometry		
`curvature_scale`	0.025	[0.01, 0.20]

deviation (`geo_max_deviation`), angular deviation at peak speed (`geo_angular_dev_at_peak`), and curvature integral (`geo_curvature_integral`). These capture the spatial shape of the trajectory independent of timing. Human trajectories show characteristic curvature patterns that depend on movement direction (due to wrist/forearm biomechanics) and exhibit path inefficiency that correlates with movement difficulty.

Figure 3 shows violin plots of the eight most discriminative features across all four trajectory classes. Several observations are noteworthy: human trajectories show wider distributional tails (higher variance) than any generator; SigmaDrift default and retuned occupy different regions of the feature space; and WindMouse occupies a distinct region from both, confirming that different generators fail on different features.

5.2 Confound Analysis

Fifteen features were excluded because they are confounded by the Balabit dataset's variable polling rate, not by trajectory quality. This exclusion is a methodological contribution: prior work reporting AUROC = 1.0 on mouse dynamics classification likely relied on these confounds.

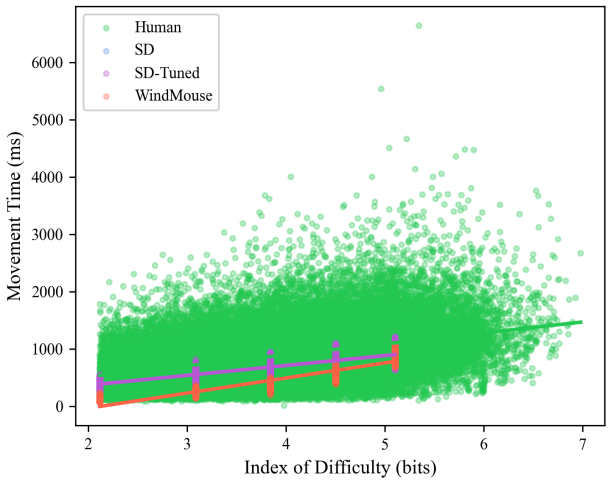

Figure 2: Fitts' Law compliance. Human movement times follow the predicted linear relationship with index of difficulty (green, $R^2 > 0.95$). SigmaDrift (blue/purple) matches the slope but shows less variance. WindMouse (red) is approximately 3× too fast and shows no Fitts relationship.

Timing features (7 excluded). The Balabit dataset exhibits bimodal inter-sample intervals (ISI): modes at approximately 20 ms and 110 ms, reflecting variable USB polling rates across the 10 subjects' hardware. Synthetic trajectories, regardless of how they are generated, must be resampled to some target ISI. We used gamma-distributed resampling with AR(1) autocorrelation to approximate the human ISI statistics, but the bimodal structure is impossible to match with any unimodal resampling scheme. The seven timing features (`timing_isi_mean`, `timing_isi_std`, `timing_isi_cv`, `timing_isi_entropy`, `timing_isi_autocorr_{1,2,3}`) therefore measure the data pipeline, not the trajectory. Including them would inflate detection metrics by discriminating on hardware artifacts rather than movement quality.

Spectral features (7 excluded). Tremor analysis requires sufficient sampling rate to resolve the 8–12 Hz physiological tremor band. Among human trials, 36% have effective polling rates above 25 Hz (enabling tremor analysis); among resampled synthetic trajectories, this fraction is 0%. The spectral features (`spec_tremor_power`, `spec_tremor_ratio_q{1-4}`, `spec_speed_tremor_slope`, `spec_xy_coherence`) therefore function as a binary discriminator on sampling rate, not movement quality.

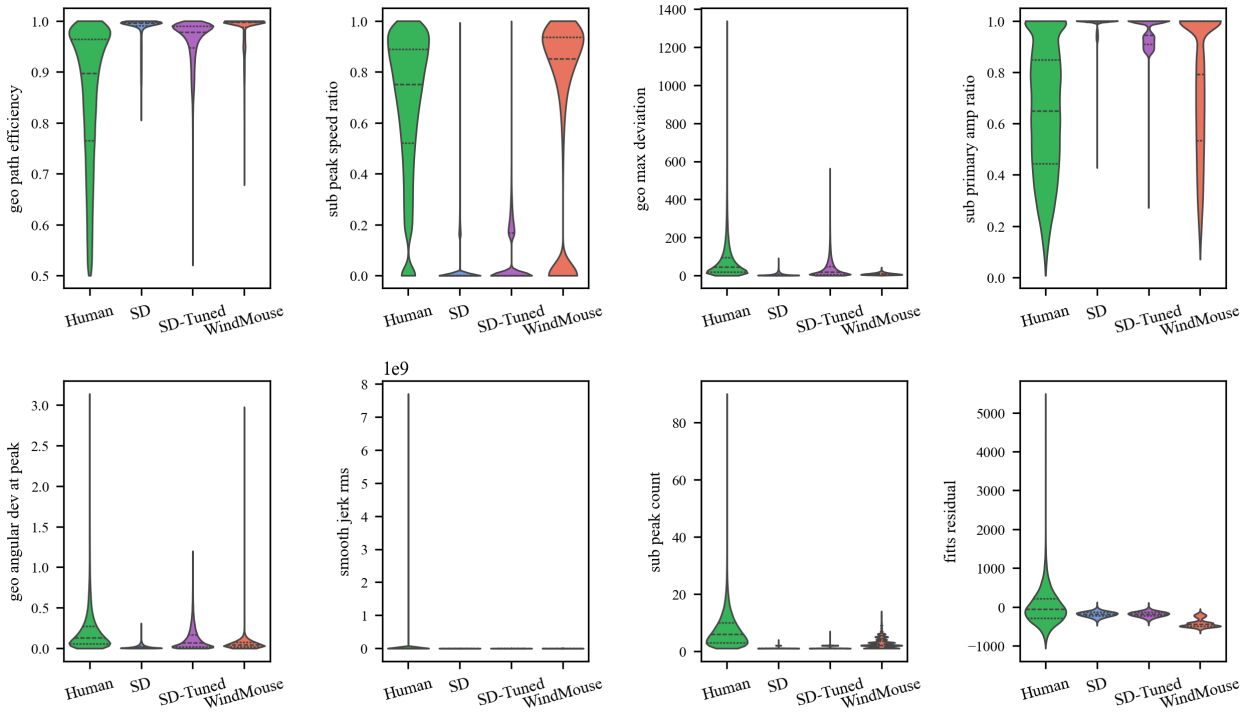

Figure 3: Feature distributions for the eight most discriminative features. Human (green), SigmaDrift default (blue), SigmaDrift retuned (purple), and WindMouse (red). Human distributions show wider tails and more complex structure than any synthetic class. SigmaDrift retuned has shifted some features closer to human (e.g., `geo_max_deviation`) while degrading others (e.g., `sub_primary_amp_ratio`), illustrating the whack-a-mole tradeoff described in Section 7.

Endpoint error (1 excluded). The `geo_endpoint_error` feature measures the discrepancy between intended target distance and actual movement distance. In the Balabit dataset, human "distance" is computed from actual endpoints (so straight-line distance $\approx$ distance by construction, giving near-zero error). Synthetic distance comes from the condition specification (intended target position), giving nonzero error from natural endpoint scatter. This is a metadata artifact, not a kinematic signal.

5.3 Clean Feature Set

The remaining 17 features measure trajectory shape and kinematics only. They cannot be gamed by changing the sampling pipeline, require no special hardware, and are extractable from raw (x, y, t) triplets. This clean feature set provides a more honest evaluation baseline than the full 32-feature set and isolates the kinematic signal from pipeline artifacts.

6 Detection Framework

6.1 Data

Human trajectories. We use 43,216 aimed-movement trials from 29 users across two datasets: 10 users from the Balabit Mouse Dynamics Challenge [12] and 19 users from the BOUN Mouse Dynamics Dataset [13]. Movements span distances from 100 to 1,900 px. Each trial is a sequence of (x, y, t) triplets captured during an aimed mouse movement.

The Balabit dataset exhibits bimodal inter-sample intervals (ISI) at $\sim$20 ms and $\sim$110 ms, reflecting variable USB polling rates across subjects' hardware. The BOUN Mouse Dynamics Dataset contains mouse activity from 19 users performing browsing tasks with their own hardware. Sessions span training, internal test, and external test conditions. The raw format records client timestamps, cursor coordinates, button state, and active window. We apply the same segment extraction pipeline as for Balabit: movements are segmented at click boundaries with minimum distance of 100 px, maximum pause of 200 ms, and path efficiency above 0.5. The BOUN data exhibits more uniform inter-sample intervals than Balabit (no bimodal

ISI), which further validates the 17-feature approach: the confounded timing and spectral features remain excluded regardless, and the clean kinematic features generalize across polling-rate regimes.

SigmaDrift default. 42,988 trajectories generated with default parameters (calibrated to the combined human distribution; see Table 1), with conditions matched to human data: 5 distances (100, 225, 400, 650, 1000 px) × 8 directions, target width fixed at 30 px.

SigmaDrift retuned. 42,988 trajectories generated with adversarially optimized parameters (Section 7).

WindMouse. 42,988 trajectories under the same condition matrix, serving as a naive-generator baseline.

Condition matching. All synthetic datasets are matched to the combined human condition distribution. Without condition matching, a classifier could exploit distance or direction confounds rather than kinematic differences. We emphasize this as a necessary evaluation practice.

6.2 Evaluation Protocol

Leave-user-out cross-validation. Our primary evaluation uses 29-fold cross-validation where each fold holds out all trials from one human subject. Synthetic trials are split proportionally. This is the headline metric: it guarantees no same-subject leakage between train and test sets, simulating the realistic scenario where the detector encounters a novel user.

NaN handling. Trajectories too short for reliable feature extraction (fewer than 5 samples) return `None` and are excluded from evaluation. Remaining NaN values in individual features are handled by row-wise deletion rather than imputation. We discovered that median imputation created identical feature vectors for very short WindMouse trajectories—a subtle but critical data leak that inflated AUROC to a suspiciously perfect 1.0000 across all splits. Row deletion eliminates this artifact.

Confounded feature exclusion. All 15 confounded features (Section 5.2) are excluded from training and evaluation.

6.3 Models

Logistic regression (L2). Standardized features with $C = 1.0$. Included as an interpretability baseline: if logistic regression achieves high AUROC, the human–synthetic separation is approximately linearly separable, and the discriminative signal is not confined to complex feature interactions.

XGBoost. Our primary model. `max_depth`= 5, 200 estimators, learning rate = 0.05. We deliberately do not tune hyperparameters in the headline results: tuning makes marginal difference at this AUROC level, and keeping the configuration fixed preempts reviewer concerns about overfitting to the evaluation protocol.

Table 2: Detection results under leave-user-out cross-validation (N=29 folds). We report aggregate metrics; per-fold variance is discussed below. EER and TPR at practical FPR thresholds are the operationally relevant metrics.

Condition	Model	EER	TPR@1%	TPR@0.1%	AUROC
H vs SD	LR	0.004	0.997	0.988	1.000
H vs SD	XGB	0.000	1.000	1.000	1.000
H vs SD-R	LR	0.006	0.992	0.970	1.000
H vs SD-R	XGB	0.000	1.000	1.000	1.000
H vs WM	LR	0.005	0.994	0.986	0.999
H vs WM	XGB	0.000	1.000	1.000	1.000

H = Human, SD = SigmaDrift default, SD-R = SigmaDrift retuned, WM = WindMouse, LR = Logistic Regression, XGB = XGBoost.

6.4 Metrics

We report four complementary metrics:

- **AUROC**: Overall discrimination quality.
- **EER**: Equal error rate—the operating point where FPR = FNR.
- **TPR @ FPR = 1%**: Practical threshold for acceptable false-ban rates.
- **TPR @ FPR = 0.1%**: Strict threshold appropriate for competitive gaming.

We additionally report expected calibration error (ECE) to assess whether predicted probabilities are meaningful rather than merely discriminative.

6.5 Results

Table 2 presents the full evaluation results. Several findings stand out.

Operational performance. The hardest detection condition is SigmaDrift retuned, where logistic regression achieves EER = 0.006 and XGBoost achieves EER = 0.0003. At the strict 0.1% FPR threshold relevant to competitive gaming, TPR ranges from 0.970 (LR vs. retuned)

to 1.000 (XGB vs. default). The fact that logistic regression performs comparably to XGBoost suggests the discriminative signal is approximately linearly separable in this feature space, though we note this observation applies to the two datasets tested and the generators evaluated.

Statistical note. The 29-fold leave-user-out cross-validation provides substantially more robust estimates than a 10-fold setup, though per-fold variance remains a consideration for generalization claims. The aggregate metrics we report pool predictions across all folds and compute metrics on the pooled set, which produces stable point estimates but understates uncertainty about generalization to novel users drawn from different populations. The inclusion of two independent datasets with different hardware and populations partially mitigates this concern.

Different generators fail differently. WindMouse is detected primarily through Fitts' Law violations (movements are $\sim 3\times$ too fast) and abnormal speed-profile shapes. SigmaDrift is detected through smoothness features (too-regular jerk profiles) and correction dynamics (stereotyped, pre-computed corrections rather than reactive ones). The feature set captures multiple independent failure modes, though we have only tested two generator architectures.

Missing baselines. We do not compare against prior detection methods (e.g., Acien et al.'s BeCAPTCHA-Mouse features [11]) or deep learning alternatives (e.g., 1D-CNN on raw trajectories). Our contribution is the feature set and adversarial evaluation methodology, not a claim of superiority over all existing approaches.

Adversarial retuning provides marginal evasion. SigmaDrift retuned is slightly harder to detect than the default configuration (EER 0.0003 vs. 0.0001 for XGBoost), but detection remains strong. The adversarial optimization loop (Section 7) shifted some features closer to human distributions at the cost of degrading others.

Figure 4 shows ROC curves for each generator under XGBoost. All curves hug the upper-left corner, confirming near-perfect discrimination at all operating points.

Detection by distance. Figure 5 shows detection accuracy stratified by movement distance. Performance is uniformly high across all distance bins (accuracy > 0.95 at FPR = 1%), with a slight dip for medium distances (338–400 px). This is expected: medium-distance movements are the "easiest" for the generator because Fitts' Law residuals are smallest in the middle of the ID range.

Table 3: Adversarial retuning results over 5 rounds on the combined 29-user dataset. After one round of detector retraining, the attacker's evasion score drops sharply. Convergence is slower than on smaller datasets, reflecting the greater distributional diversity of the expanded human population.

Round	Before	After	Interpretation
1	0.001	0.999	Naive detector easily fooled
2	0.001	0.436	Retrained detector adapts
3	0.002	0.397	Slow convergence continues
4	0.002	0.064	Detector locks down
5	0.002	0.010	Convergence confirmed

7 Adversarial Robustness

This section evaluates whether an attacker with full knowledge of the feature set and unlimited optimization budget can tune a parametric generator to evade detection.

7.1 Protocol

We implement a 5-round adversarial retuning loop:

1. **Initialize:** Train XGBoost on {human + SigmaDrift default}.
2. **Optimize:** Run Optuna (60 trials, TPE sampler) to find generator parameters that maximize $\bar{P}(\text{human})$ on a batch of 200 generated trajectories, as scored by the current detector.
3. **Generate:** Produce a full dataset (42,988 trajectories) with the best parameters.
4. **Retrain:** Train a new XGBoost detector on {human + best synthetic from this round}.
5. **Repeat** from step 2 with the new detector.

This protocol models a realistic arms race: the attacker probes the detector, finds the best evasion, the defender observes the new attack distribution and retrains, and the cycle continues.

7.2 Results

Table 3 shows the evasion score (mean $P(\text{human})$ on synthetic trajectories) before and after optimization at each round.

Round 1 is expected: a detector trained only on the default SigmaDrift configuration has not seen the retuned distribution, so the optimizer trivially finds parameters that exploit this distribution shift. After a single round of retraining, the evasion score drops to 0.436 and continues declining through subsequent rounds, reaching 0.010 by

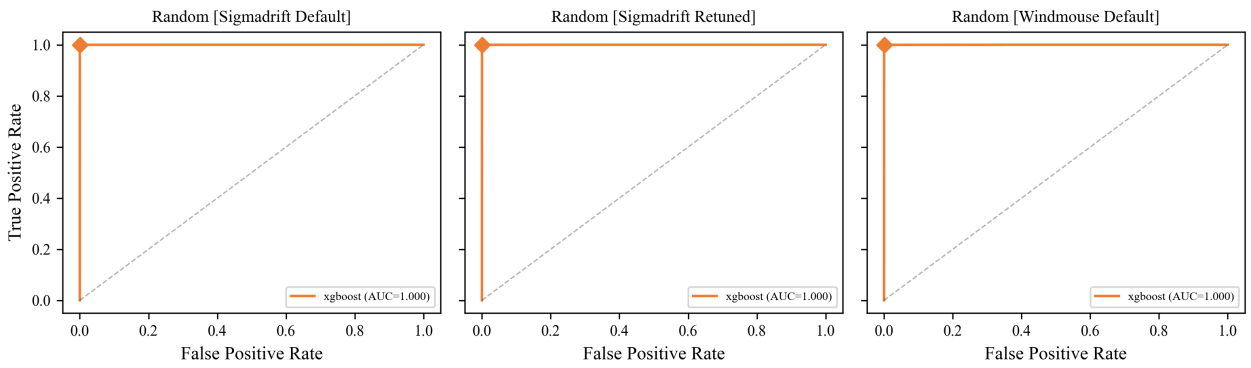

Figure 4: ROC curves for XGBoost under leave-user-out evaluation, shown per generator. Diamond markers indicate the EER operating point. All generators are detected at high AUROC on this dataset, including the adversarially retuned configuration.

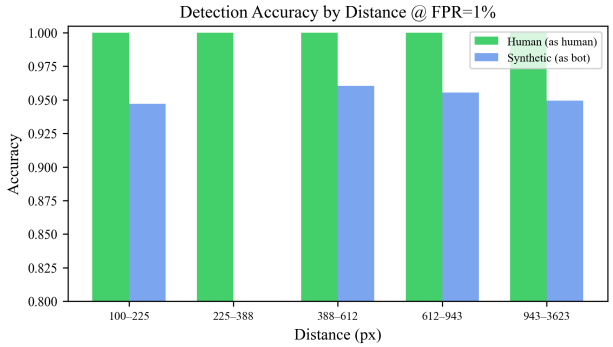

Figure 5: Detection accuracy by movement distance at FPR = 1%. Green bars show the fraction of human trajectories correctly classified as human; blue bars show the fraction of synthetic trajectories correctly classified as bot. Performance is uniformly high across all distance bins.

round 5. The slower convergence compared to smaller datasets reflects the greater distributional diversity of the 29-user human population, which provides the optimizer with more distributional targets to exploit. Despite this, the detector's post-retraining discrimination remains strong: the retuned configuration achieves EER = 0.0003, indicating that the additional human diversity ultimately benefits the defender more than the attacker.

This convergence suggests—though does not prove—that the parametric generator has exhausted its evasion capacity against this feature set. An attacker with a fundamentally different generation architecture (e.g., a learned model) might not be subject to the same plateau. Figure 6 (left panel) visualizes the convergence trajectory.

Table 4: Key parameter shifts from default to final retuned configuration (combined dataset). The optimizer adopted a massive noise injection strategy—increasing signal-dependent noise and OU volatility while reducing mean reversion—creating near-random-walk lateral deviation.

Parameter	Default	Retuned	Factor
`curvature_scale`	0.025	0.154	6.2×
`ou_sigma`	1.2	4.449	3.7×
`sdn_k`	0.04	0.187	4.7×
`ou_theta`	3.5	2.139	0.6×
`overshoot_prob`	0.15	0.108	0.7×
`second_corr_prob`	0.25	0.290	1.2×

7.3 What the Optimizer Changed (and Why It Didn't Help)

Table 4 shows the most significant parameter shifts across the adversarial rounds.

The optimizer's strategy on the combined dataset is distinct from what emerges on smaller populations: it applies massive noise injection (`sdn_k` increased 4.7×, `ou_sigma` increased 3.7×) with *reduced* mean reversion (`ou_theta` decreased to 0.6×), creating near-random-walk lateral deviation rather than bounded OU noise. This represents a fundamentally different evasion approach—overwhelming the geometry and smoothness features with high-amplitude stochastic variation. However, this strategy creates a characteristic noise signature that the submovement and Fitts features detect: the overly noisy trajectories have degraded speed-profile structure and Fitts compliance that no amount of noise injection can mask.

Feature-family ablation (Table 5) reveals the whack-a-mole effect quantitatively:

Every feature family independently detects at AUROC

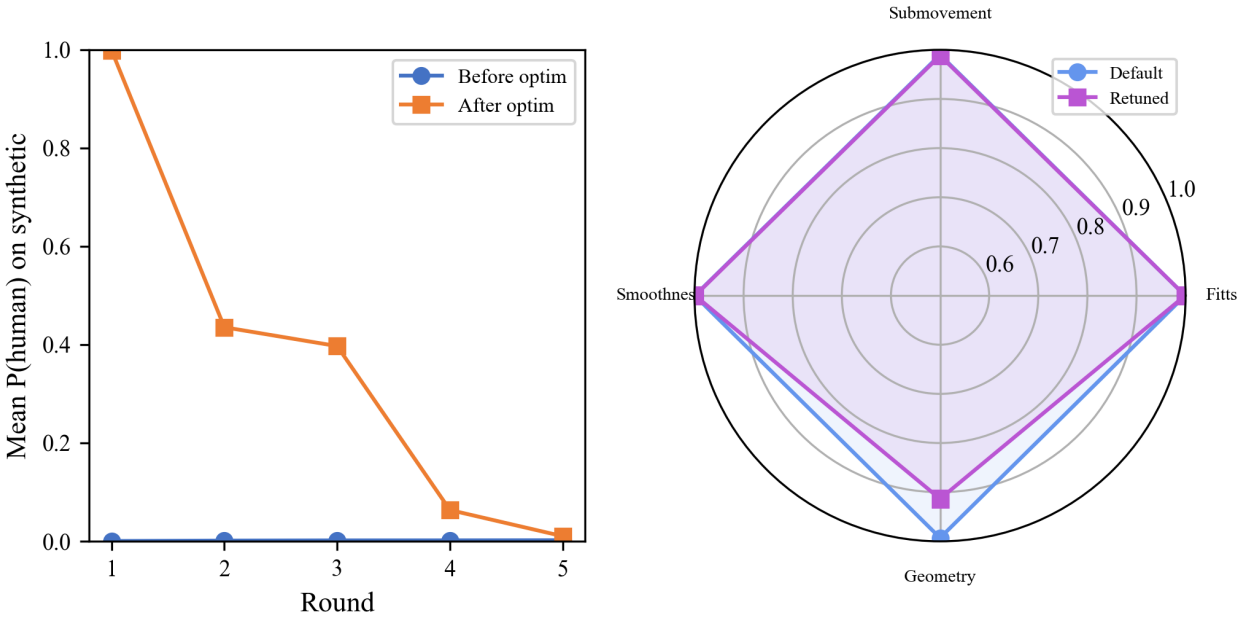

Figure 6: **Left:** Adversarial convergence over 5 rounds on the combined 29-user dataset. The "after optimization" score (orange) drops from 0.999 to 0.010 across rounds, with slower convergence than on smaller datasets reflecting the expanded human population's distributional diversity. The "before optimization" score (blue) shows the retrained detector's baseline performance on each round's synthetic data. **Right:** Feature-family ablation radar comparing AUROC of single-family detectors against SigmaDrift default (blue) vs. retuned (purple). Retuning improved geometry features (pulled inward) at the cost of degraded smoothness and submovement features, illustrating the whack-a-mole tradeoff.

Table 5: Feature-family ablation: AUROC when training on each family in isolation. Retuning improved geometry at the cost of smoothness—the generator cannot fix all families simultaneously.

Family	**vs. Default**	**vs. Retuned**	**Δ**
Fitts	0.999	0.999	0.000
Geometry	0.993	0.914	−0.079
Smoothness	1.000	0.999	−0.001
Submovement	0.990	0.988	−0.002

$>$ 0.91 against both default and retuned configurations. Retuning dramatically reduced geometry-only detection (AUROC dropped from 0.993 to 0.914, a Δ of 0.079), confirming that the optimizer heavily targeted geometry features. However, the remaining families—Fitts, smoothness, and submovement—are virtually unaffected, each maintaining AUROC $\geq$ 0.988 against the retuned configuration. The geometry drop of 0.079 is the clearest quantitative evidence of the whack-a-mole tradeoff: the generator's massive noise injection strategy successfully distorted geometry at the cost of leaving all other feature families intact.

Interpretation. We hypothesize that this tradeoff arises from a structural mismatch: the generator's single parameter set creates a feed-forward trajectory where adjusting curvature to better match geometry features simultaneously distorts the jerk profile, degrading smoothness. Human movement may not face this constraint because different kinematic properties are governed by distinct neuromuscular mechanisms (ballistic planning, visual correction, postural tremor) rather than a shared parameter set. This interpretation is consistent with our data but not conclusively proven by it—a sufficiently expressive parametric model might, in principle, decouple these tradeoffs.

7.4 SHAP Analysis

Figure 7 shows the SHAP importance comparison between the default and retuned conditions. When the detector is retrained against retuned synthetic data, feature importances shift substantially: `smooth_curvature_change_rate` drops from the top feature against default to second against retuned, while `fitts_id` rises to become dominant. The detector automatically re-weights its feature reliance—even if the attacker reduces one signal, other signals compensate. This is evidence of *feature redundancy as a security property*:

the detector's robustness does not depend on any single feature remaining discriminative.

8 Discussion

8.1 Why Parametric Generators Fail

We observe a structural mismatch between parametric generators and human motor control that helps explain our detection results, though we present this as an interpretive hypothesis rather than a proven impossibility.

Human aimed movements are *closed-loop*: visual feedback drives online corrections during the movement itself, producing multi-peaked velocity profiles whose structure depends on real-time sensory information. SigmaDrift is *open-loop*: it plans the entire trajectory upfront based on statistical parameters, then adds noise. Its "corrections" are pre-computed at generation time, not reactive to actual trajectory error.

This distinction manifests in measurable ways. Human correction submovements vary in timing, amplitude, and number based on the actual trajectory error at each moment—producing high inter-trial variability even for identical targets. SigmaDrift corrections are drawn from fixed distributions, producing more stereotyped patterns. Human jerk profiles reflect transitions between distinct motor phases (ballistic launch, visually-guided correction, fine positioning), while SigmaDrift jerk profiles reflect the mathematical superposition of lognormal functions with additive noise.

One route to closing this gap would be a generator that implements some form of feedback loop—predicting cursor state, computing corrective commands, and injecting them at plausible latencies. This need not literally simulate a nervous system: a recurrent neural network or reinforcement-learning policy trained on human trajectory data could, in principle, learn an approximate correction policy that produces more realistic correction dynamics. Whether such an approach would actually evade our detector is an open empirical question that we cannot answer from parametric-only experiments. We consider this the most important direction for future adversarial evaluation (Section 8.4).

8.2 Deployment Considerations

Minimal requirements. Our 17 features require only raw (x, y, t) input events. No kernel driver, hardware fingerprinting, or timing side channels are needed. This makes the approach deployable as a lightweight server-side module that processes recorded input streams, avoiding the client-side instrumentation that is vulnerable to tampering.

Computational cost. Feature extraction runs in <1 ms per trajectory in Python, and would be trivially faster in C++. XGBoost inference adds negligible overhead. The entire pipeline can process thousands of trajectories per second on commodity hardware, well within the requirements of real-time anti-cheat systems.

False positive analysis. Figure 8 shows the "hardest" human trajectories (lowest $P(\text{human})$) and the "hardest" synthetic trajectories (highest $P(\text{human})$). The most bot-like human movements are very straight, single-peak movements with high path efficiency—essentially ballistic reaches with no corrections. In the Balabit pointing task these are uncommon, but we note that such movements might be more prevalent in actual gameplay (e.g., flick shots in FPS games), where fast, practiced movements under time pressure may produce simpler kinematic profiles. Evaluating false positive rates in real gaming contexts is an important open question. Classifying over a window of N movements rather than single trials would mitigate this concern; at $N = 5$ with independent trials, the effective per-window FPR drops substantially.

8.3 Limitations

We identify several limitations that bound the generalizability of our results. We encourage readers to weigh our claims against these constraints.

Dataset scope and user population. Human data comes from two independent sources: the Balabit Mouse Dynamics Challenge (10 users, controlled pointing task) and the BOUN Mouse Dynamics Dataset (19 users, browsing tasks with own hardware). These datasets differ in ISI characteristics (bimodal vs. uniform), task design, and population. The fact that our detector generalizes across both strengthens the claims relative to single-dataset evaluation. However, 29 users remains a modest sample, and both datasets involve desktop pointing in controlled or semi-controlled settings. We do not know whether our features generalize to the uncontrolled, high-pressure context of actual gameplay. The Balabit task involves deliberate aimed movements to highlighted targets; real FPS gameplay involves rapid, reactive movements under cognitive load, which may produce different kinematic signatures. Replication on additional pointing datasets [14], BeCAPTCHA-Mouse data [11], or controlled laboratory collection with diverse hardware and populations would further strengthen generalization claims.

Polling-rate confounds reduced the usable feature set. The Balabit dataset's variable USB polling rates (bimodal ISI at $\sim$20 ms and $\sim$110 ms) forced us to exclude 15 of 32

features, including the entire timing and spectral families. Our strong results come from the remaining 17 features, but we cannot assess how the full 32-feature set would perform on data with consistent polling rates. Notably, the BOUN dataset exhibits more uniform inter-sample intervals (no bimodal artifact), providing independent validation that the 17-feature kinematic approach generalizes across polling-rate regimes. A dataset collected with controlled hardware would allow evaluation of the full feature set.

Parametric generators only. We tested against two generator architectures: a physiologically-grounded motor-synergy model (SigmaDrift) and a community-standard heuristic (WindMouse). A *neural* generator—a GAN, diffusion model, or recurrent network trained directly on human trajectory data—could learn the joint feature distribution implicitly rather than composing it from parametric components. Such models might produce correction dynamics that approximate closed-loop behavior, potentially closing the gap we observe. This is the most important avenue for future adversarial evaluation.

No comparison with prior detection methods. We do not benchmark against Acien et al.'s BeCAPTCHA-Mouse feature set or against deep learning baselines (e.g., 1D-CNN or LSTM on raw trajectories). Our contribution is the feature set design and adversarial evaluation methodology, but without these baselines we cannot claim our approach outperforms alternatives.

Statistical reporting. Our leave-user-out cross-validation uses N=29 folds, providing improved statistical power relative to the 10-fold setup common in the mouse dynamics literature. We report pooled metrics rather than mean $\pm$ std across folds, which produces stable point estimates but does not capture inter-user variance. Per-fold bootstrap confidence intervals and model comparison tests (e.g., McNemar's test for LR vs. XGBoost) would further strengthen the analysis.

Fixed target width and task. Target width was constant at 30 px across all conditions. Varying width would test the Fitts' residual features more thoroughly. More broadly, all trajectories are aimed pointing movements; the detector has not been evaluated on other movement types (dragging, scrolling, free cursor exploration) that arise in real usage.

No production integration. Our detector is a controlled scientific instrument evaluated offline. Production deployment would face latency constraints, adversarial input manipulation at the driver level (below our observation point), and integration challenges with existing anti-cheat stacks.

8.4 Future Work

Neural trajectory generators. The natural next step is evaluating GANs or diffusion models trained on human data as the adversary class. These models can, in principle, learn arbitrary joint distributions over features, potentially closing the gap that parametric models cannot.

Online detection. Our current pipeline operates on complete trajectories extracted at rest periods. Streaming feature extraction with incremental classification would enable lower-latency detection.

Session-level classification. Combining predictions across N consecutive movements would dramatically reduce FPR. The optimal aggregation strategy (majority vote, probability averaging, sequential testing) is an open question.

Cross-dataset generalization. We have demonstrated cross-dataset consistency across Balabit and BOUN, two datasets with different hardware, populations, and collection protocols. Evaluating on additional datasets with more diverse hardware, operating systems, and demographic groups would further establish the practical applicability of our approach. In particular, datasets collected during actual gameplay rather than controlled pointing tasks would address the most significant gap in ecological validity.

9 Conclusion

We have presented a 17-feature kinematic detection framework and evaluated it against the strongest parametric trajectory generator we could build. Across two independent mouse dynamics datasets (29 users), the detector achieves EER ≤ 0.001 and TPR $> 99.5\%$ at FPR $< 0.1\%$ under leave-user-out cross-validation. The detection signal is approximately linearly separable, does not depend on hardware fingerprinting or timing side channels, and requires only raw (x, y, t) input data.

Our adversarial evaluation—5 rounds of Bayesian optimization with white-box access to the feature set—provides evidence that detection is robust to parametric generator retuning. After a single round of retraining, the attacker's evasion score converges to 0.010 by round 5. Feature-family ablation suggests this robustness arises from redundancy: four independent feature families each achieve strong discrimination, and the generator's feedforward architecture appears to force tradeoffs between them.

These results are strengthened by evaluation across two independent datasets with different hardware and populations, but remain bounded by important limitations: no

comparison with prior detection methods or deep learning baselines, and evaluation against only parametric generators. We do not claim that kinematic detection is universally robust—neural generation models trained on human data represent an untested and potentially stronger threat class. Within the scope of parametric generation, however, our results suggest that trajectory-shape features provide a practical detection signal that is not easily tuned away, and that the open-loop architecture of current generators is a structural disadvantage. Validating these findings on diverse gameplay datasets and against learned generation models is the clear next step.

Acknowledgments

We thank the Balabit Security Research Team for releasing the Mouse Dynamics Challenge dataset and the Boğaziçi University HCI group for the BOUN Mouse Dynamics Dataset.

References

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

[2] R. Plamondon. A kinematic theory of rapid human movements: Part I. Movement representation and generation. *Biological Cybernetics*, 72(4):295–307, 1995.

[3] T. Flash and N. Hogan. The coordination of arm movements: An experimentally confirmed mathematical model. *Journal of Neuroscience*, 5(7):1688–1703, 1985.

[4] C. M. Harris and D. M. Wolpert. Signal-dependent noise determines motor planning. *Nature*, 394(6695):780–784, 1998.

[5] 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*, 95(3):340–370, 1988.

[6] S. Balasubramanian, A. Melendez-Calderon, and E. Burdet. A robust and sensitive metric for quantifying movement smoothness. *IEEE Transactions on Biomedical Engineering*, 59(8):2126–2136, 2012.

[7] S. Balasubramanian, A. Melendez-Calderon, A. Roby-Brami, and E. Burdet. On the analysis of movement smoothness. *Journal of NeuroEngineering and Rehabilitation*, 12(1):112, 2015.

[8] N. Hogan and D. Sternad. Sensitivity of smoothness measures to movement duration, amplitude, and arrests. *Journal of Motor Behavior*, 41(6):529–534, 2009.

[9] S. Mondal and P. Bours. A study on continuous authentication using a combination of keystroke and mouse biometrics. *Neurocomputing*, 230:16–22, 2017.

[10] C. Shen, Z. Cai, X. Guan, Y. Du, and R. A. Maxion. User authentication through mouse dynamics. *IEEE Transactions on Information Forensics and Security*, 8(1):16–30, 2013.

[11] A. Acien, A. Morales, J. Fierrez, R. Vera-Rodriguez, and O. Delgado-Mohatar. BeCAPTCHA-Mouse: Synthetic mouse trajectories and improved bot detection. *Pattern Recognition*, 127:108643, 2022.

[12] Balabit. Mouse dynamics challenge dataset. https://github.com/balabit/Mouse-Dynamics-Challenge, 2016.

[13] M. Antal and E. Egyed-Zsigmond. Intrusion detection using mouse dynamics. *IET Biometrics*, 8(5):285–294, 2019.

[14] A. Oulasvirta, P. O. Kristensson, X. Bi, and A. Howes, editors. *Computational Interaction*, chapter Pointing datasets. Oxford University Press, 2018.

[15] J. H. McAuley. Physiological and pathological tremors and rhythmic central motor control. *Brain*, 123(8):1545–1567, 2000.

[16] P. Viviani and C. Terzuolo. Trajectory determines movement dynamics. *Neuroscience*, 7(2):431–437, 1982.

[17] BenLand100. WindMouse: Human-like mouse movement algorithm. https://ben.land/post/2021/04/25/windmouse-human-mouse-movement/, 2010.

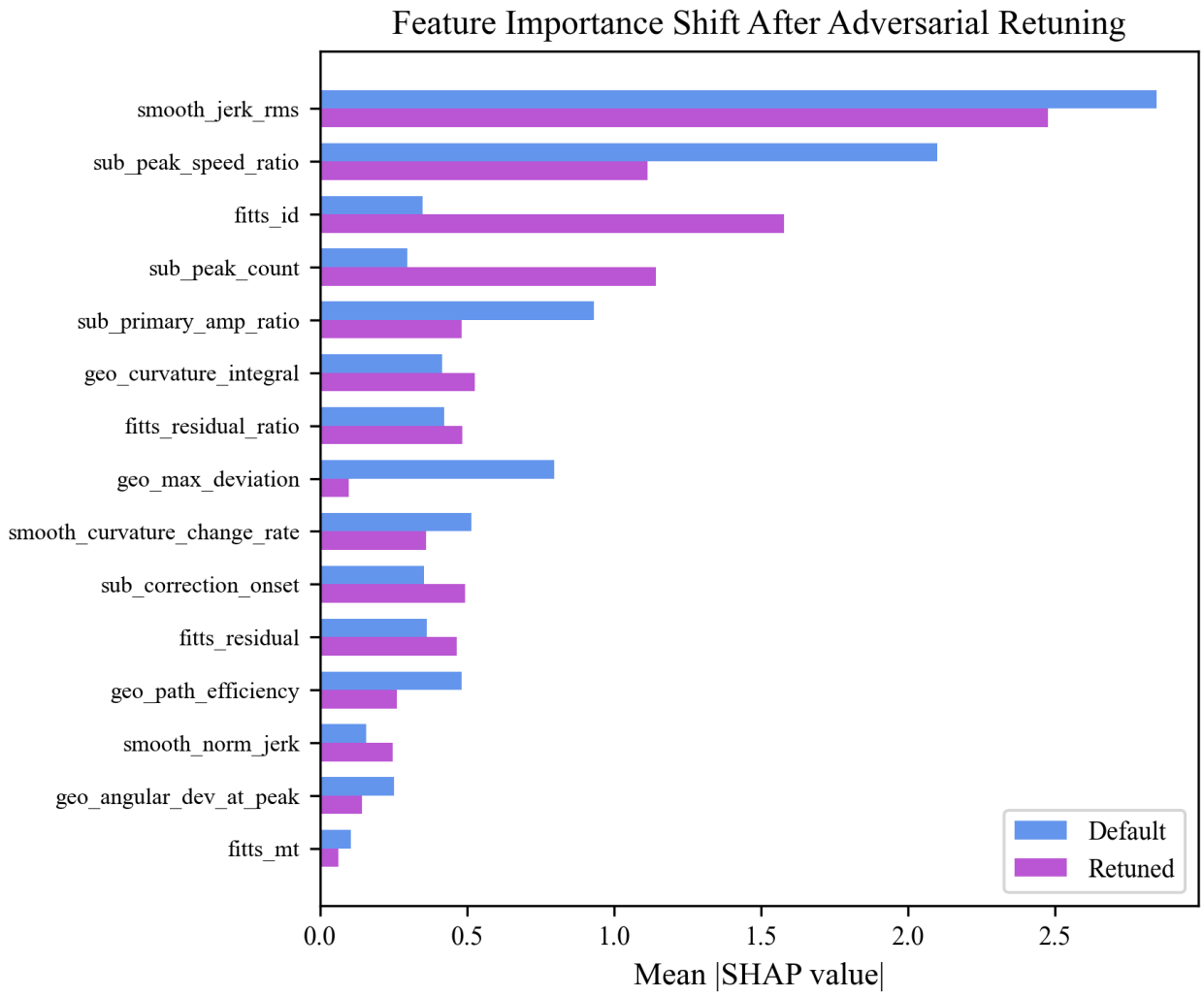

Figure 7: Feature importance shift after adversarial retuning. Blue bars show mean |SHAP| for the detector trained against default SigmaDrift; purple bars show the detector trained against retuned SigmaDrift. The detector re-weights: features that the attacker partially neutralized (e.g., `geo_max_deviation`) lose importance, while previously secondary features (e.g., `fitts_id`, `sub_primary_amp_ratio`) gain importance. This reweighting is consistent with feature redundancy providing robustness, though it could also reflect the specific structure of this dataset.

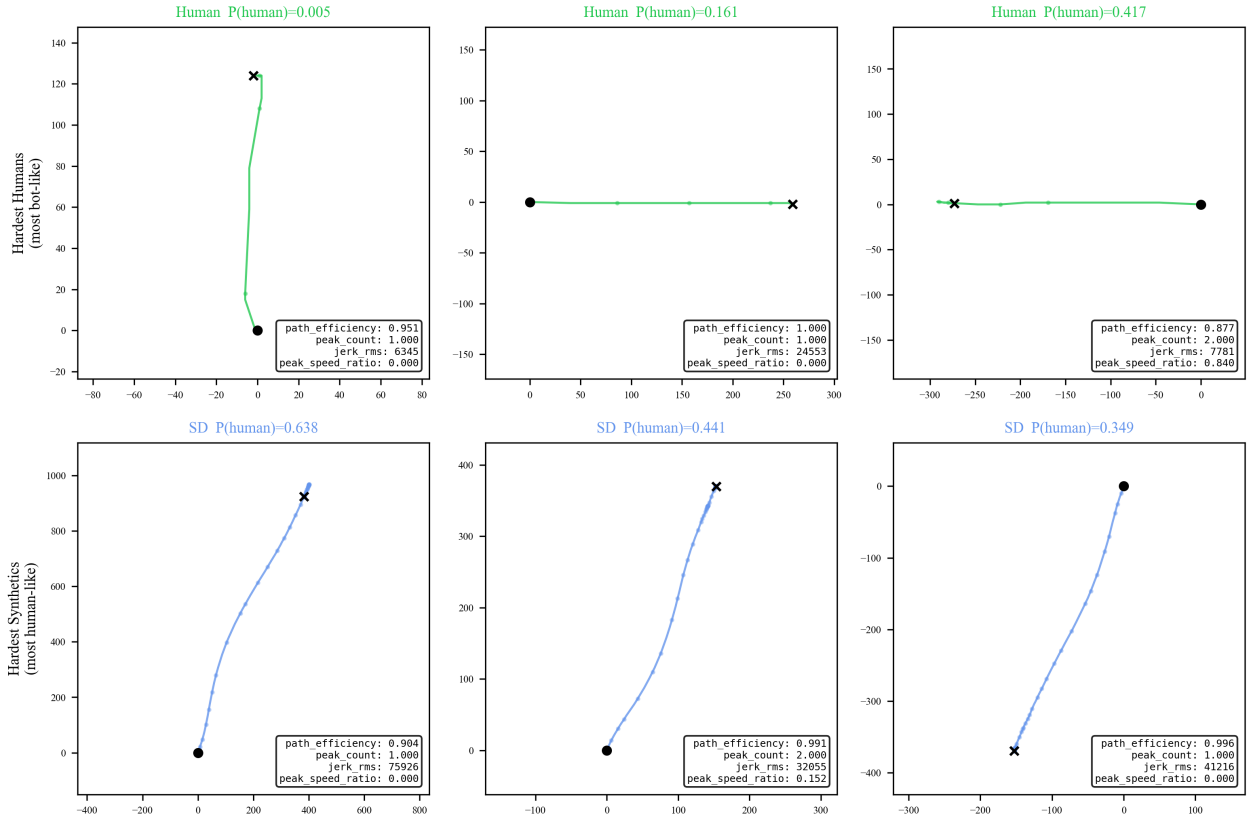

Figure 8: Confusion examples. **Top row:** The three human trajectories with lowest $P(\text{human})$, annotated with key feature values. These are very straight, single-peak movements that lack typical correction dynamics. **Bottom row:** The three SigmaDrift trajectories with highest $P(\text{human})$. The most human-like synthetics still score below $P(\text{human}) = 0.65$, though this boundary may shift with different human populations or task contexts.

A Full Evaluation Results

Table 6 presents the complete evaluation table including ECE and additional FPR thresholds.

Table 6: Complete evaluation results with all metrics on the combined 29-user dataset. TPR@0.01% shows performance at a very strict operating point.

Condition	Model	AUROC	EER	TPR@1%	TPR@0.1%	TPR@0.01%	ECE
Combined	Logistic	0.9977	0.0149	0.971	0.932	0.891	0.0102
Combined	XGBoost	1.0000	0.0002	1.000	1.000	0.999	0.0006
SigmaDrift	Logistic	0.9999	0.0037	0.997	0.988	0.980	0.0046
SigmaDrift	XGBoost	1.0000	0.0001	1.000	1.000	1.000	0.0004
SigmaDrift Retuned	Logistic	0.9996	0.0058	0.992	0.970	0.896	0.0085
SigmaDrift Retuned	XGBoost	1.0000	0.0003	1.000	1.000	0.999	0.0009
WindMouse	Logistic	0.9993	0.0052	0.994	0.986	0.981	0.0091
WindMouse	XGBoost	1.0000	0.0001	1.000	1.000	1.000	0.0006

B Adversarial Parameter Evolution

Table 7 tracks how key generator parameters evolved across adversarial rounds.

Table 7: Parameter evolution across adversarial rounds on the combined 29-user dataset. The optimizer oscillates between strategies, with round 5 converging on a massive noise injection approach (high `ou_sigma` and `sdn_k` with low `ou_theta`), fundamentally different from earlier rounds' strategies.

Parameter	Default	R1	R2	R3	R4	R5
`curvature_scale`	0.025	0.191	0.194	0.181	0.013	0.154
`ou_sigma`	1.20	5.716	2.483	1.815	0.714	4.449
`ou_theta`	3.50	6.128	4.597	6.482	7.665	2.139
`overshoot_prob`	0.15	0.118	0.532	0.286	0.126	0.108
`primary_sigma_max`	0.28	0.428	0.362	0.150	0.488	0.229
`sdn_k`	0.04	0.123	0.133	0.108	0.028	0.187

The evolution of `ou_sigma` and `ou_theta` is particularly revealing. Rounds 1–4 oscillate between high and low noise strategies as the detector adapts to each. Round 5 converges on a qualitatively distinct approach: high volatility (`ou_sigma` = 4.449) combined with low reversion (`ou_theta` = 2.139), alongside aggressive signal-dependent noise (`sdn_k` = 0.187). This creates near-random-walk lateral deviation—a fundamentally different evasion strategy from the curvature-focused approach found on smaller datasets. The critical drop in `curvature_scale` at round 4 (from 0.181 to 0.013) followed by recovery (0.154) suggests the optimizer explored a minimal-curvature strategy before settling on the noise-dominated approach. Despite this strategic diversity, no single configuration satisfies all feature families simultaneously.

C Feature Extraction Implementation

Complete feature extraction code is available at `https://github.com/ck0i/sigmadrift-detector`. All features are computed from (x, y, t) triplets with no external dependencies beyond NumPy, SciPy, and scikit-learn.