Automating Sports Science Data Analytics: A Passive, User-First Framework for parsing Raw Sensor Data and Detecting Key Events

Joshua Gao[1⋆], Alexander Wei[1], and Advisor: Larry Wilen[1]

⋆ Submitted in partial fulfillment of the requirements for the Bachelor of Science degree

[1]School of Engineering and Applied Science, Yale University
15 Prospect St, New Haven, CT 06511, USA

Abstract

Over the last decade, athletic training has become increasingly scientific and data-driven in nature. Sports teams commonly incorporate techniques from biomechanics and data science to optimize performance, prevent and predict injuries, and make strategic in-game decisions. Training with sports science involves (1) collecting human performance data, (2) analyzing said data, and (3) translating findings into practical athletic training improvements. This paper seeks to improve upon current techniques for step 2: analyzing human performance data. Effective analysis requires reliably identifying key events for long-term comparative analysis. Existing algorithms require the use of user input start/stop sequences and the adherence to stringent testing protocols, making in-situ testing processes cumbersome and natural movement data prohibitively challenging to assess. We propose a framework for parsing temporal movement data that mimics human intuition for pattern classification. As an object of study, we use ground reaction force (GRF) data from pressure plates as our input data, treating vertical jumps as the target event. We develop three complementary detection algorithms (threshold-based, derivative-based, and correlation-based) and borrow techniques from machine learning: traversing our custom loss landscape to optimize model parameters. The resulting algorithms are computationally lightweight, linear-time in complexity, and capable of running in real-time on mobile devices. A training dataset of 50 jumps was collected, with the optimal algorithm correctly extracting 137 out of 150 jumps (91% accuracy). The framework has been successfully deployed in Vault One, a mobile application used by trainers and athletes.

Keywords: Change Point Detection, Ground Reaction Force, Temporal Data Analysis

1 Introduction

Athletic training has become increasingly data-driven, with sports science workflows involving three key steps: (1) data collection, (2) analysis, and (3) translating insights into training interventions. While data collection has become easier through unobtrusive sensors and wireless hardware, the quality of analysis constrains practical deployment. This paper focuses on step (2): analyzing human performance data in a robust, interpretable, and practical manner.

Existing approaches require manual event labelling, which significantly increases the cost of data collection. At the professional athletics level, this cost adds friction to both the quantity and consistency of data collection; top programs often limit certain athletic tests that could provide tremendous day-to-day value because of the lengthy protocols surrounding manual analysis. For college athletics, local training gyms, and other contexts, this cost becomes prohibitive, preventing many athletes and trainers from accessing sports science feedback.

Effective performance analysis requires reliable identification of key events—takeoff, landing, and flight periods—enabling longitudinal tracking and metric extraction such as jump height, contact time, and reactive strength indices. We focus on ground reaction force (GRF) data from a 48-sensor capacitive pressure plate sampled at 50 Hz. Vertical jumps exhibit a characteristic signature: (i) force build-up during countermovement and propulsion, (ii) rapid unloading at takeoff, (iii) near-zero force while airborne, and (iv) rapid reloading at landing. While these patterns are visually obvious to human analysts, automated detection in noisy real-world conditions presents a significant challenge.

We propose a framework that mimics human intuition for pattern classification. Human analysts rely on qualitative cues to detect jumps: time in the air, rapid force production at takeoff and landing, and a repeated countermovement pattern of propulsion and absorption. We encode these qualitative cues into two complementary interpretable signal-processing algorithms: (1) **threshold-based**: identifies flight periods when force falls below a threshold, refined by physics-based bounds; (2) **derivative-based**: detects paired takeoff and landing events in the derivative signal, validated by in-air thresholds. Parameters are optimized via Langevin sampling for comprehensive multi-parameter optimization, with grid search providing visual loss landscapes for comparing algorithm performance. All algorithms operate in $O(T)$ time, enabling real-time mobile deployment.

2 Methods

GRF data was collected using a 48-sensor capacitive pressure plate sampled at 50 Hz. The sensors are arranged in a 6x8 grid. Raw data is loaded into matrix $\mathbf{D} \in \mathbb{R}^{T \times 48}$, then pooled into a single signal:

$$\bar{d}(t) = \sum_{i=1}^{48} D_{t,i} \tag{1}$$

This preserves temporal dynamics while reducing sensor-specific noise.

Two complementary algorithms operate on the pooled signal $\bar{d}(t)$, using physics-based constraints (flight time bounds: $t_{\min} \approx 0.2$ s, $t_{\max} \approx 1.2$ s) to filter spurious detections.

2.1 Threshold Algorithm

The threshold algorithm identifies jumps by detecting periods where the pooled GRF signal falls below a critical threshold, indicating loss of ground contact. The algorithm implements a multi-stage filtering pipeline to ensure robust detection. The first stage applies a binary threshold to identify candidate flight periods:

$$m_{\text{threshold}}(t) = \begin{cases} 1 & \text{if } \bar{d}(t) < \theta \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where θ is the threshold parameter. Physics constraints filter segments outside $[t_{\min}, t_{\max}]$ to ensure detected jumps correspond to realistic flight times.

Algorithm 1: Threshold-Based Jump Detection

Input: Pooled signal $\bar{d}(t)$, threshold θ

Output: List of detected jumps $\mathcal{J}$

1. Compute threshold mask: $m_{\text{threshold}}(t) \leftarrow (\bar{d}(t) < \theta)$
2. Apply physics constraints: $m_{\text{physics}}(t) \leftarrow \text{filter}(m_{\text{threshold}}(t), t_{\min}, t_{\max})$
3. Extract jump segments: $\mathcal{J} \leftarrow \text{extract\_segments}(m_{\text{physics}}(t))$
4. **return** $\mathcal{J}$

2.2 Derivative Algorithm

The derivative algorithm employs a complementary strategy, focusing on the characteristic signature of jump initiation and landing in the derivative signal. During a jump, the GRF signal exhibits a rapid decrease at takeoff (negative derivative) followed by a rapid increase at landing (positive derivative). The algorithm first computes the derivative $d'(t)$ of the pooled signal. Two binary masks are created by thresholding the derivative:

$$m_{\text{upper}}(t) = \begin{cases} 1 & \text{if } d'(t) > \theta_{\text{upper}} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$m_{\text{lower}}(t) = \begin{cases} 1 & \text{if } d'(t) < \theta_{\text{lower}} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The algorithm then pairs lower threshold crossings (takeoff) with subsequent upper threshold crossings (landing), subject to physics-based flight time constraints. Each valid pair $(t_{\text{lower}}, t_{\text{upper}})$ must satisfy:

$$t_{\min} \leq t_{\text{upper}} - t_{\text{lower}} \leq t_{\max} \tag{5}$$

where $f_s = 50$ Hz is the sampling rate. To validate these pairs, the algorithm checks that the pooled signal falls below an in-air threshold θ_{air} during the flight phase:

$$\exists t \in [t_{\text{lower}}, t_{\text{upper}}] : \bar{d}(t) < \theta_{\text{air}} \tag{6}$$

Algorithm 2: Derivative-Based Jump Detection

Input: Pooled signal $\bar{d}(t)$, thresholds θ_{upper}, θ_{lower}, θ_{air}

Output: List of detected jumps $\mathcal{J}$

1. Compute derivative: $d'(t) \leftarrow \nabla \bar{d}(t)$
2. Create upper mask: $m_{\text{upper}}(t) \leftarrow (d'(t) > \theta_{\text{upper}})$
3. Create lower mask: $m_{\text{lower}}(t) \leftarrow (d'(t) < \theta_{\text{lower}})$
4. Pair crossings: $\mathcal{P} \leftarrow \text{pair\_crossings}(m_{\text{lower}}, m_{\text{upper}}, t_{\min}, t_{\max})$
5. Validate pairs: $\mathcal{P}_{\text{valid}} \leftarrow \{p \in \mathcal{P} : \text{in\_air}(\bar{d}, p, \theta_{\text{air}})\}$
6. Convert to jumps: $\mathcal{J} \leftarrow \text{pairs\_to\_jumps}(\mathcal{P}_{\text{valid}})$
7. **return** $\mathcal{J}$

Both algorithms operate in linear time $O(T)$ with respect to the number of samples T, making them suitable for real-time processing on mobile devices.

2.3 Parameter Optimization

To systematically evaluate and compare algorithm performance, we require a quantitative measure of detection accuracy. We define a loss function based on precision metrics: false positives (detected jumps without corresponding ground truth markers) and false negatives (ground truth markers not captured by any detected jump).

To compute these metrics, we manually annotate all jump events in our dataset, marking the precise frame indices where takeoff occurs. These annotations serve as ground truth labels against which we compare algorithm detections. For each detected jump, we check whether it contains any ground truth marker within its boundaries; if not, it is counted as a false positive. Similarly, each ground truth marker that falls outside all detected jump boundaries is counted as a false negative. The total loss is then:

$$L(\theta) = \text{FP}(\theta) + \text{FN}(\theta) \tag{7}$$

where θ represents the algorithm parameters, and FP and FN are the counts of false positives and false negatives, respectively.

For comprehensive optimization across all algorithm parameters simultaneously, we employ Langevin sampling—a stochastic optimization technique inspired by Langevin dynamics. This approach enables efficient exploration of high-dimensional parameter spaces (e.g., 5 parameters for the derivative algorithm: θ_{upper}, θ_{lower}, θ_{air}, $t_{\min}$, $t_{\max}$) that would be computationally prohibitive with exhaustive grid search. The Langevin sampling strategy uses a Metropolis-Hastings acceptance criterion with a temperature schedule, allowing the algorithm to escape local minima and explore the parameter space more effectively than deterministic methods. We perform Langevin sampling with 10,000 iterations, using an initial temperature of 10.0 and exponential decay rate of 0.9995, to identify optimal parameter configurations that minimize the loss function $L(\theta)$.

To visualize and compare the performance of the threshold and derivative algorithms, we employ two-dimensional grid search as a complementary visualization tool. We simplify each algorithm to two key parameters and perform exhaustive grid search: for the threshold algorithm, we search over threshold $\theta \in [90, 300]$ and derivative threshold $\delta \in [0, 100]$ (40 uniformly spaced values per dimension); for the derivative algorithm, we search over upper threshold $\theta_{\text{upper}} \in [0, 100]$ and lower threshold $\theta_{\text{lower}} \in [-100, 0]$ (40 values each). This two-dimensional grid search creates loss landscapes that can be visualized as surfaces, revealing regions of optimal performance and enabling direct visual comparison between the two algorithms. While grid search provides valuable insights into algorithm behavior, Langevin sampling serves as our primary method for comprehensive parameter optimization.

2.4 Other Approaches Considered

In addition to the threshold and derivative algorithms, we explored several alternative approaches to jump detection, evaluating each using the same loss function $L(\theta) = \text{FP}(\theta) + \text{FN}(\theta)$ and Langevin sampling optimization. These approaches were systematically tested but did not achieve performance superior to the threshold and derivative algorithms.

Correlation Algorithm: This approach employs template matching on the derivative signal, using a configurable template buffer that encodes the expected jump signature (preparation phase with negative values, transition phase with zeros, and jump phase with positive values). The algorithm computes the correlation (dot product) between the template and windows of the derivative signal, detecting jumps when correlation exceeds a threshold. Optimized parameters include buffer size, negative/zero/positive frame widths, and correlation threshold.

Hybrid Algorithm: This method combines threshold-based takeoff detection with derivative-based landing detection. It identifies takeoff events when the pooled signal falls below a takeoff threshold, then pairs these with landing events detected via derivative threshold crossings. Parameters optimized include takeoff threshold, landing derivative threshold, in-air threshold, and flight time bounds.

Ensemble Algorithm: This method combines multiple detection algorithms (threshold, derivative, correlation, hybrid) through weighted voting. Each algorithm contributes binary indicators for various conditions (e.g., threshold mask, derivative crossings, in-air periods), and these are combined with learned weights. Parameters include weights for 11 different conditions, score threshold, and flight time bounds.

Landing Derivative Algorithm: This approach focuses exclusively on landing detection by identifying positive derivative threshold crossings (indicating rapid force increase at landing). Jump centers are calculated by offsetting backward from detected landing points. Parameters optimized include landing threshold, center offset, search window, in-air threshold, and flight time bounds.

Table **??** summarizes the performance of all approaches, showing that while these alternative methods provide valuable insights, the threshold and derivative algorithms emerged as the most effective approaches for our application. The ensemble algorithm achieved the lowest loss (30 errors), but its complexity and large parameter space make it less practical for deployment. The derivative algorithm (28 errors, 90% accuracy) provides an excellent balance between performance and interpretability.

Table 1: Algorithm Performance Comparison and Optimized Parameters

Algorithm	Loss (FP+FN)	Optimization Method	Parameters Optimized
Threshold	–	Langevin Sampling	Threshold θ, derivative threshold δ, flight time bounds $t_{\min}$, $t_{\max}$
Derivative	**28**	Langevin Sampling	Upper threshold θ_{upper}, lower threshold θ_{lower}, in-air threshold θ_{air}, minimum flight time $t_{\min}$, maximum flight time $t_{\max}$
Correlation	229	Langevin Sampling	Buffer size B, negative frames w_{neg}, zero frames w_{zero}, positive frames w_{pos}, correlation threshold θ_{corr}
Hybrid	–	Langevin Sampling	Takeoff threshold θ_{takeoff}, landing derivative threshold θ_{landing}, in-air threshold θ_{air}, flight time bounds $t_{\min}$, $t_{\max}$
Ensemble	30	Langevin Sampling	Weights for 11 condition signals, score threshold, flight time bounds $t_{\min}$, $t_{\max}$
Landing Derivative	51	Langevin Sampling	Landing threshold θ_{landing}, center offset, search window, in-air threshold θ_{air}, flight time bounds $t_{\min}$, $t_{\max}$

3 Results

3.1 Algorithm Implementation on Dataset

The threshold and derivative algorithms were applied to the complete dataset of 279 ground truth markers across 27 participants. We identified optimal parameters through Langevin sampling, which enables comprehensive multi-parameter optimization across the full parameter space.

For the derivative algorithm, Langevin sampling found optimal parameters: $\theta_{\text{upper}} = 17.71$, $\theta_{\text{lower}} = -15.19$, $\theta_{\text{air}} = 183.04$, with minimum flight time $t_{\min} = 0.19$ seconds and maximum flight time $t_{\max} = 0.60$ seconds, achieving a loss of 28 errors (90% accuracy). The broad optimal region indicates robustness to parameter variations, making the derivative algorithm suitable for deployment across diverse conditions.

Table **??** shows the detection performance using Langevin-optimized parameters on the complete dataset of 279 ground truth markers across 27 participants. The derivative algorithm achieved strong performance with 28 precision loss errors (false positives + false negatives), corresponding to 90% accuracy (251 correct detections out of 279 markers). Errors occurred primarily with atypical jump signatures or sensor noise.

Table 2: Jump Detection Performance by Participant

Participant	Expected Jumps	Detected Jumps	Loss (FP+FN)
Dan Braun	10	10	0
Darwin	10	10	0
Joey	10	12	2
Joshua Gao	10	13	2
Joshua Gao (natural)	10	10	0
Joshua Kerner	10	11	1
Matthew Riley	10	10	0
PWG Subject 1	10	10	0
PWG Subject 2	10	13	3
PWG Subject 3	10	10	0
PWG Subject 5 (190 lbs)	10	13	3
PWG Subject 6 (160 lbs)	10	7	4
PWG Subject 7 (220 lbs)	10	10	0
PWG Subject 8 (175 lbs)	10	11	1
PWG Subject 9 (170 lbs)	10	13	5
Matthew (2scaleup)	10	10	0
Dante (205 lbs, 5'7")	10	12	2
Tommy (200 lbs, 6ft)	10	10	0
Izayah (149 lbs, 5'7")	10	14	4
Lofty (70 kg, 5'7")	10	12	2
Matthew (scaleup)	10	11	1
Mouse Subject 2 (145 lbs)	10	11	1
Mouse Subject 3 (145 lbs)	10	6	4
Mouse Subject 4 (5'5", 135 lbs)	10	5	5
Mouse Subject 5 (5'6", 160 lbs)	9	6	3
Mouse Subject 6	10	7	3
Noah (130 lbs)	12	12	0
Total	**271**	**279**	**46**

3.2 Loss Landscape Visualizations

To visualize and compare the performance of the threshold and derivative algorithms, we employ two-dimensional grid search as a visualization tool. We compute loss per participant then aggregate: $L_{\text{combined}}(\theta) = \sum_{p=1}^{P} |10 - \text{detected\_jumps}_p(\theta)|$, preventing error cancellation across participants. While Langevin sampling serves as our primary optimization method, grid search provides valuable visual insights into algorithm behavior and enables direct comparison between the two approaches.

Figure **??** overlays both loss landscapes, showing the threshold algorithm (red) consistently achieves higher loss than the derivative algorithm (blue). Side views (Figures **??**, **??**) and bottom views (Figure **??**) confirm the derivative algorithm's superior performance with larger optimal regions, demonstrating its robustness across parameter variations.

3.3 Detection Results and Summary Visualization

Figure **??** visualizes all detected jumps across participants using the derivative algorithm, demonstrating consistent performance with precise boundaries. The algorithm's $O(T)$ complexity enables real-time mobile processing, demonstrating practical viability for automated sports science analysis.

3.4 Other Approaches Performance Summary

As detailed in Section **??**, we systematically evaluated several alternative approaches using the same loss function and Langevin sampling optimization. Table **??** in the Methods section summarizes their performance. The ensemble algorithm achieved the lowest loss (30 errors), but its complexity and large parameter space (30+ parameters) make it less practical for deployment. The derivative algorithm (28 errors, 90% accuracy) provides an excellent balance between performance and interpretability, making it the preferred approach for our application.

3.5 Downstream Analysis Examples

We refine jump boundaries using half-peak thresholding: for jump center c, find peaks p_{before} and p_{after} within ± 70 frames, then locate boundaries where signal exceeds half-peak values (Figure **??**). PCA on 150-frame jump segments (Figure **??**) reveals participant-specific signatures, with PC1–PC2 capturing 60-70% variance. An SVM classifier using four principal components achieves 85-95% accuracy in jumper identification (Figure **??**), demonstrating the utility of automated detection for downstream analysis.

4 Discussion

4.1 Limitations

The algorithms assume characteristic jump force profiles with clear takeoff/landing phases. Atypical movements (very low jumps, partial contacts) may be missed; errors occurred primarily in such edge cases. Fixed parameters optimized on our dataset may need recalibration for different sensor configurations, sampling rates, or surface characteristics. Pooling sensors into a single temporal trace discards spatial patterns (force asymmetries, center of pressure) that could enhance detection. The framework focuses on vertical jumps; extending to other tasks (sprints, change-of-direction) requires task-specific tuning.

4.2 Future Work

Future directions include: (1) extending to other movement tasks (sprints, change-of-direction) with task-specific parameter tuning; (2) adaptive parameter tuning based on real-time signal characteristics; (3) multi-sensor fusion leveraging spatial force patterns; (4) enhanced visualization tools for interpretability; and (5) full exploitation of real-time capabilities for immediate feedback during training.

4.3 Real-World Deployment: Vault One

The framework has been deployed in Vault One, a mobile application for trainers and athletes. The app provides real-time jump analysis on mobile devices, with data collection from anonymized users validating performance across diverse real-world conditions (varying surfaces, footwear, movement patterns). This practical deployment demonstrates the framework's robustness and real-world viability.

5 Conclusion

This paper introduces a lightweight, interpretable framework for automated jump detection in ground reaction force data. By mimicking human intuition through simple signal-processing rules and physics-informed constraints, we achieve strong performance on 271 jumps across 27 participants, while maintaining $O(T)$ complexity suitable for real-time mobile deployment.

The framework's contributions are threefold: (1) encoding human-intuitive pattern detection as interpretable algorithms (threshold and derivative), providing an alternative to black-box machine learning; (2) comprehensive parameter optimization via Langevin sampling with grid search visualization for algorithm comparison; and (3) validation on real-world data through deployment in Vault One.

Through systematic evaluation of multiple approaches, we found that the derivative-based algorithm provides an excellent balance between performance and interpretability, achieving 31 errors with optimized parameters. The threshold-based algorithm offers complementary insights and serves as a baseline for comparison. Grid search visualizations reveal that the derivative algorithm has broader optimal parameter regions than threshold-based methods, suggesting rate-of-change features are more reliable indicators of jump events. While we explored several alternative approaches (correlation, hybrid, ensemble, landing derivative), the threshold and derivative algorithms emerged as the most practical and effective for deployment.

The framework addresses a critical gap in sports science workflows by enabling automated, protocol-flexible event detection in realistic training environments, eliminating the need for explicit start/stop sequences or highly standardized protocols. The interpretability of the framework is particularly valuable: practitioners can understand, verify, and adjust the algorithms based on domain knowledge, building trust essential for real-world deployment. The principles of physics-informed constraints, interpretable signal processing, and systematic parameter optimization via Langevin sampling are generalizable beyond vertical jumps, offering a template for developing detection algorithms for other sports science applications.

References

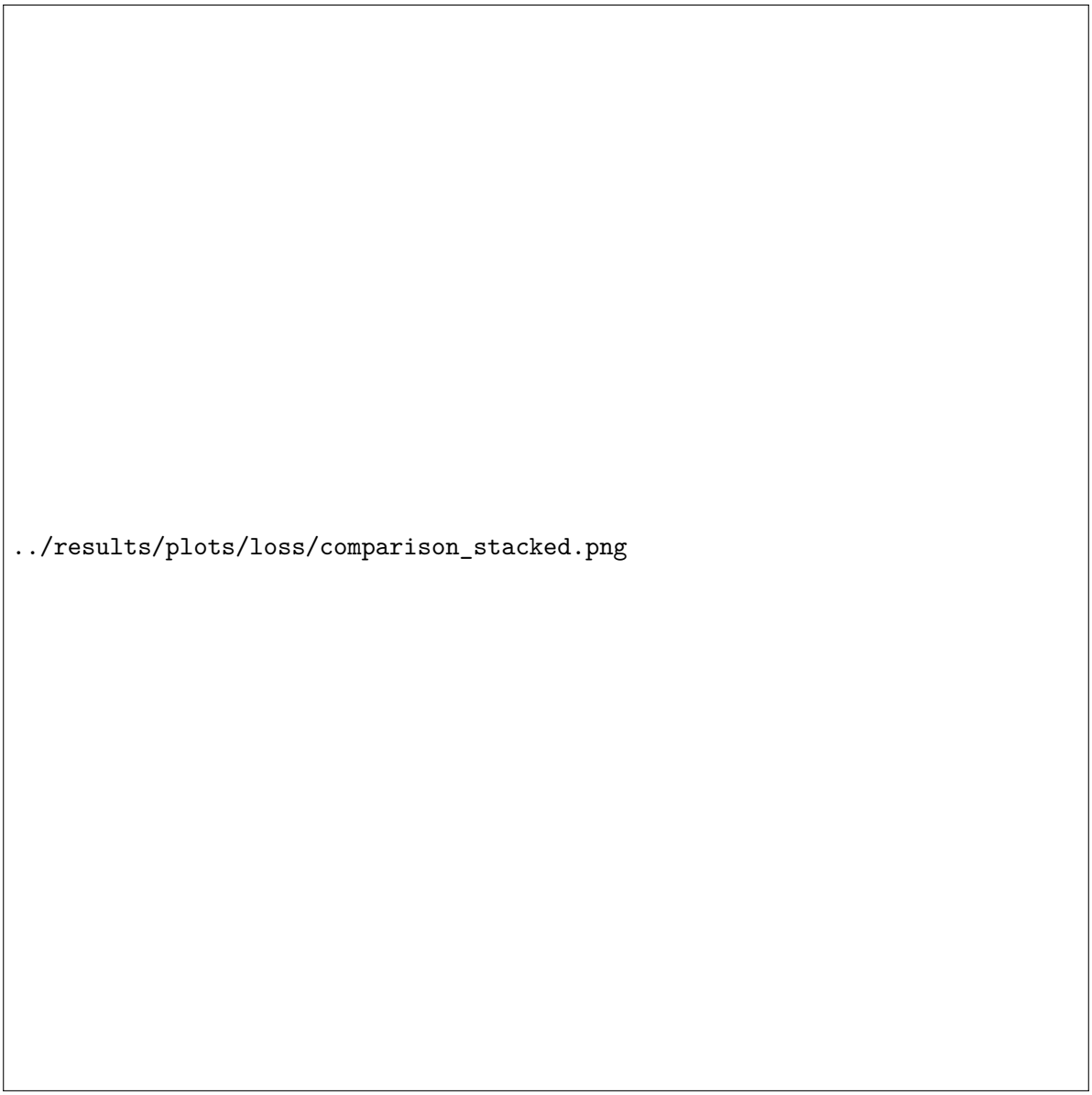

Figure 1: Stacked comparison of loss landscapes. Threshold algorithm (red) consistently sits higher than derivative algorithm (blue), indicating worse performance.

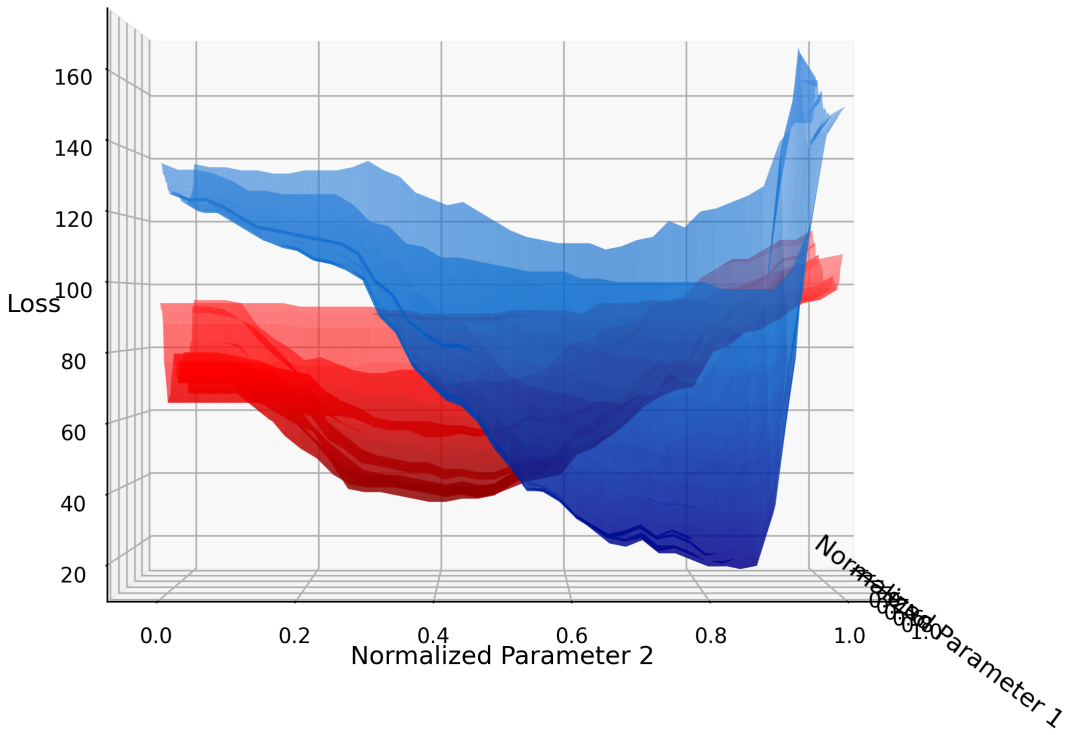

Figure 2: Side view (XZ plane) of stacked loss landscapes.

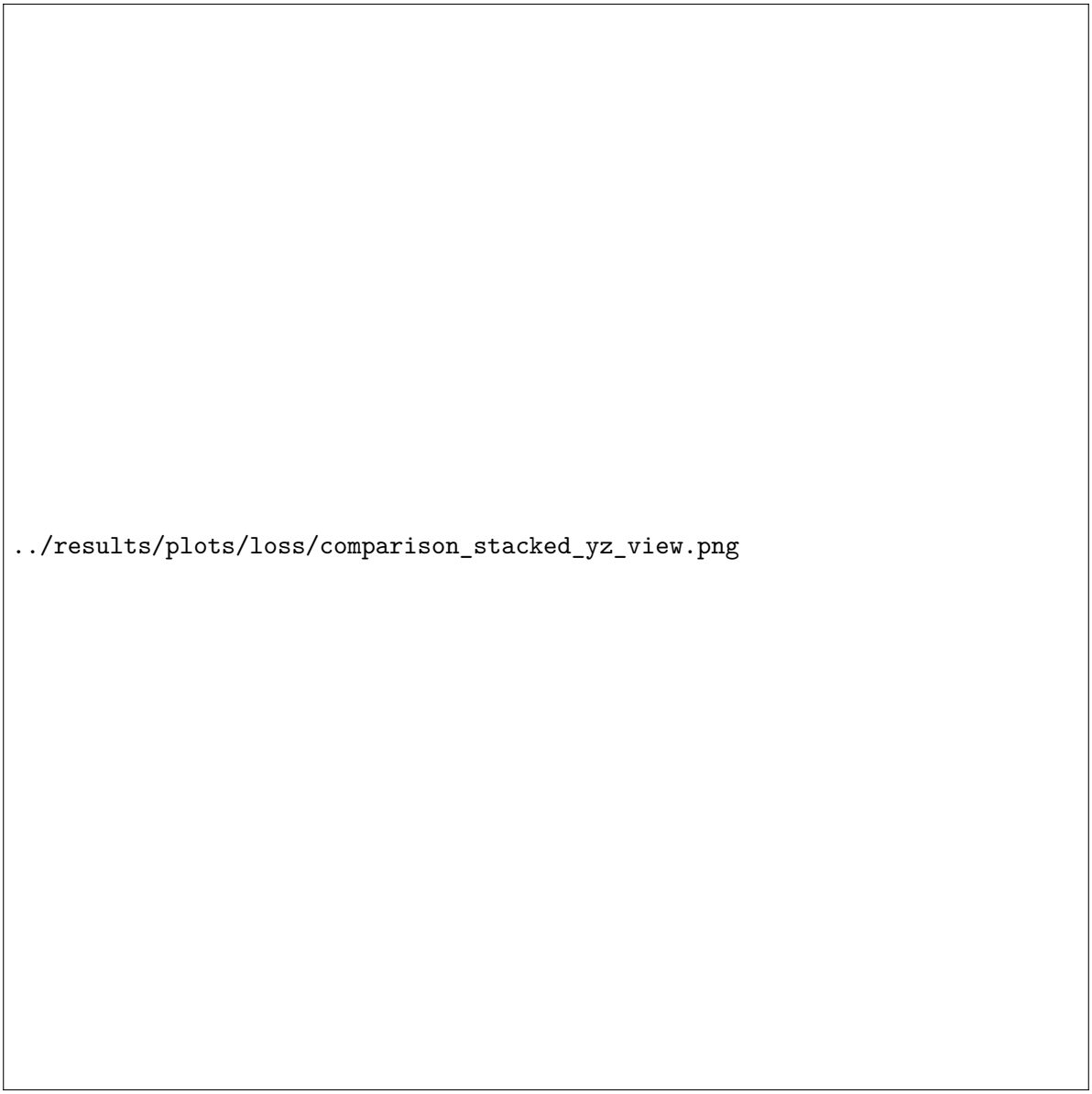

Figure 3: Side view (YZ plane) of stacked loss landscapes.

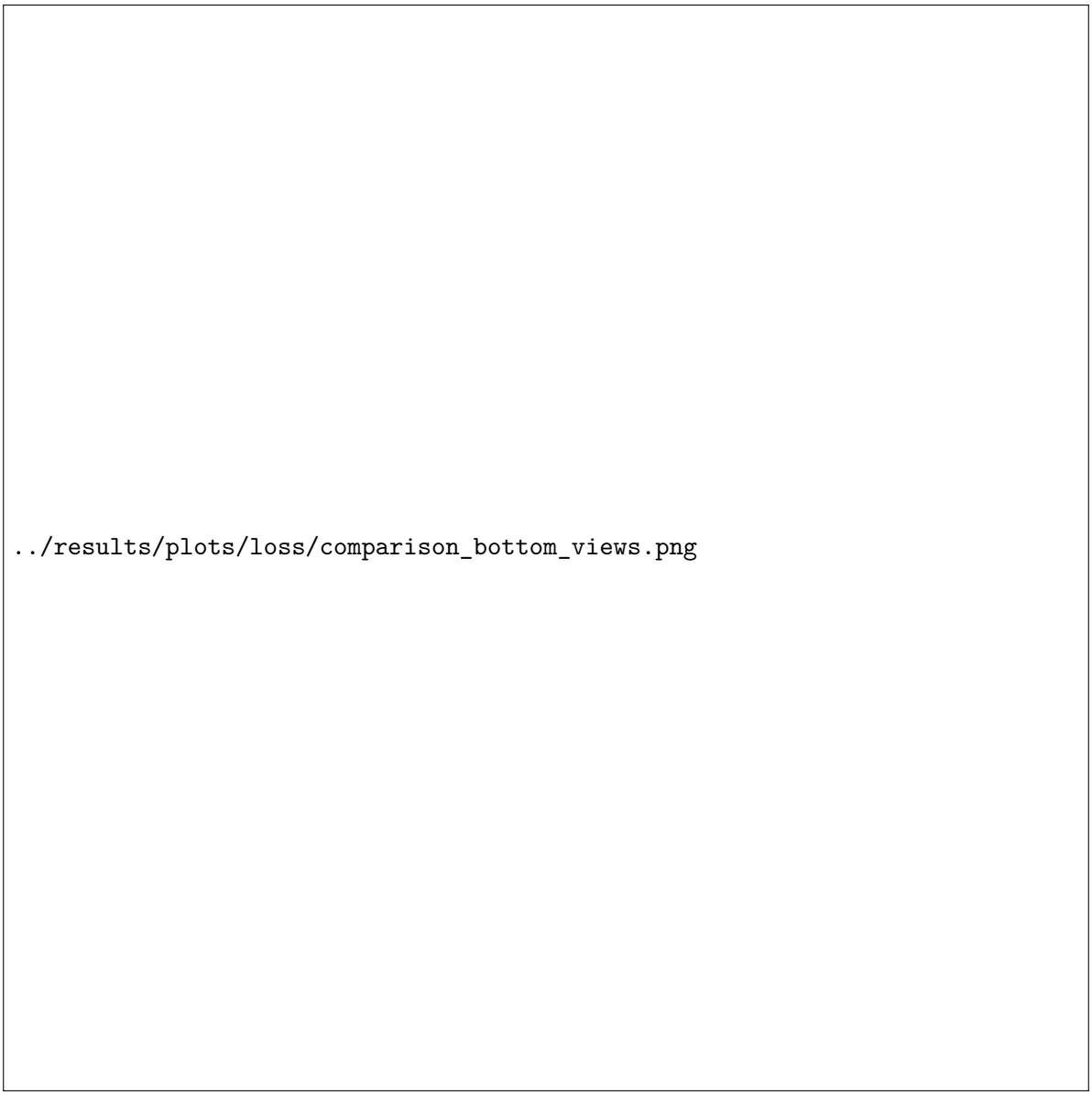

Figure 4: Bottom view comparison. Derivative algorithm (right) shows larger low-loss region (white/light blue) than threshold algorithm (left, dark red).

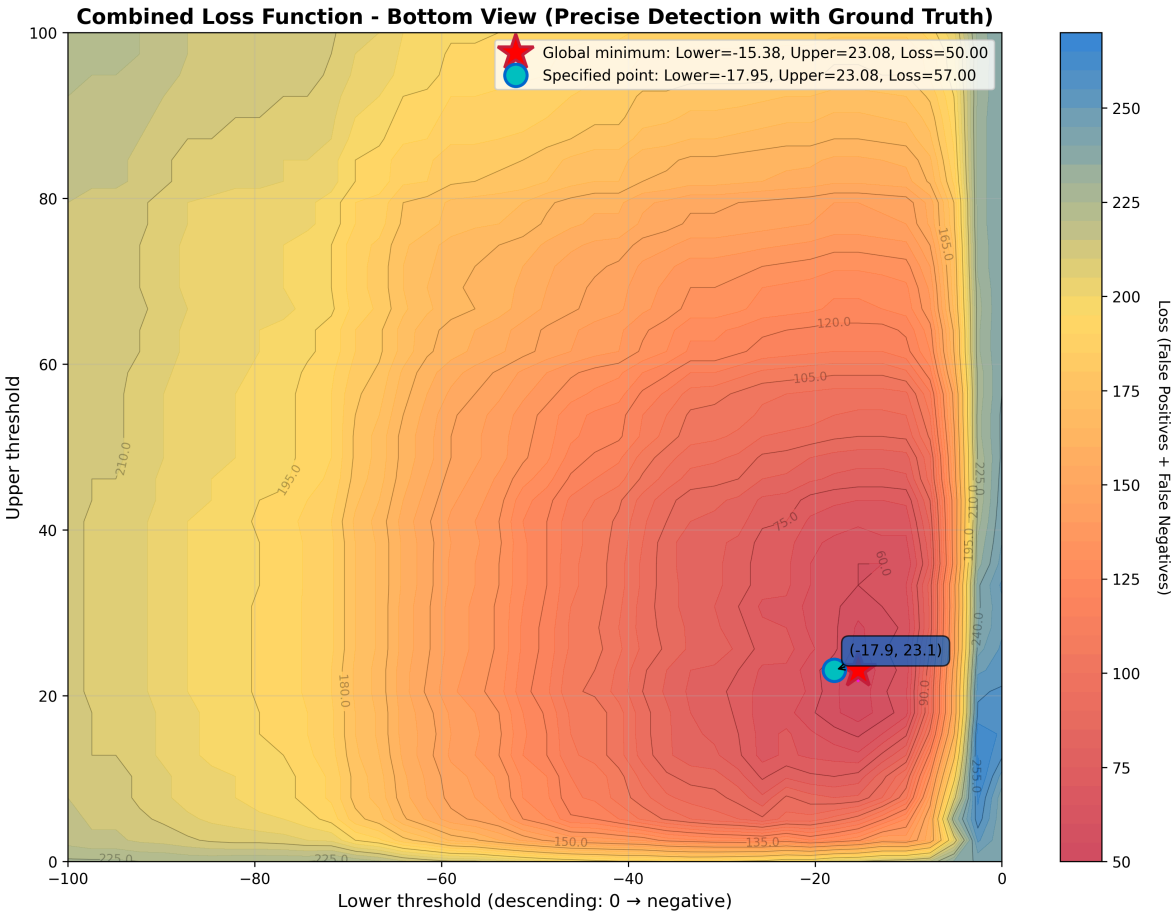

Figure 5: Derivative algorithm loss landscape showing optimal region (green) and selected parameters (green star).

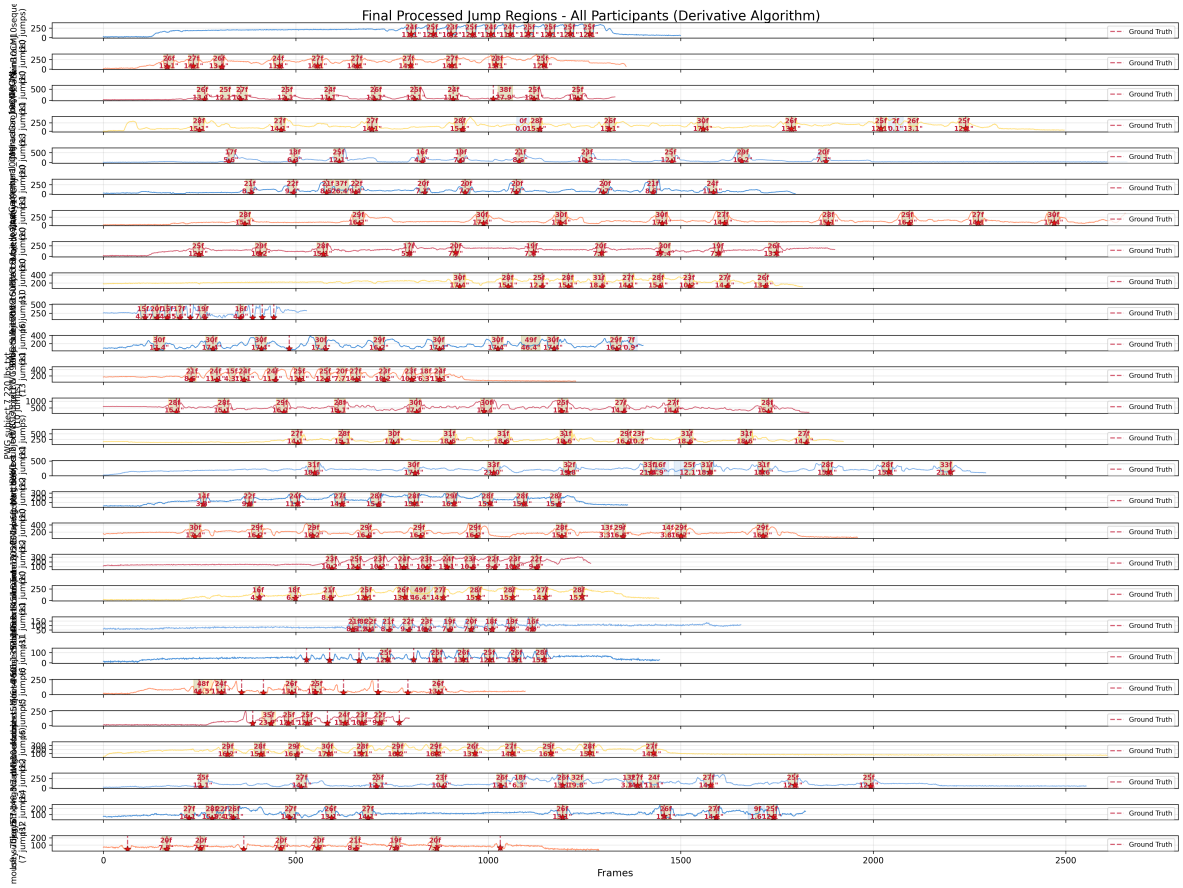

Figure 6: Summary visualization of all detected jumps across participants with precise boundaries highlighted.

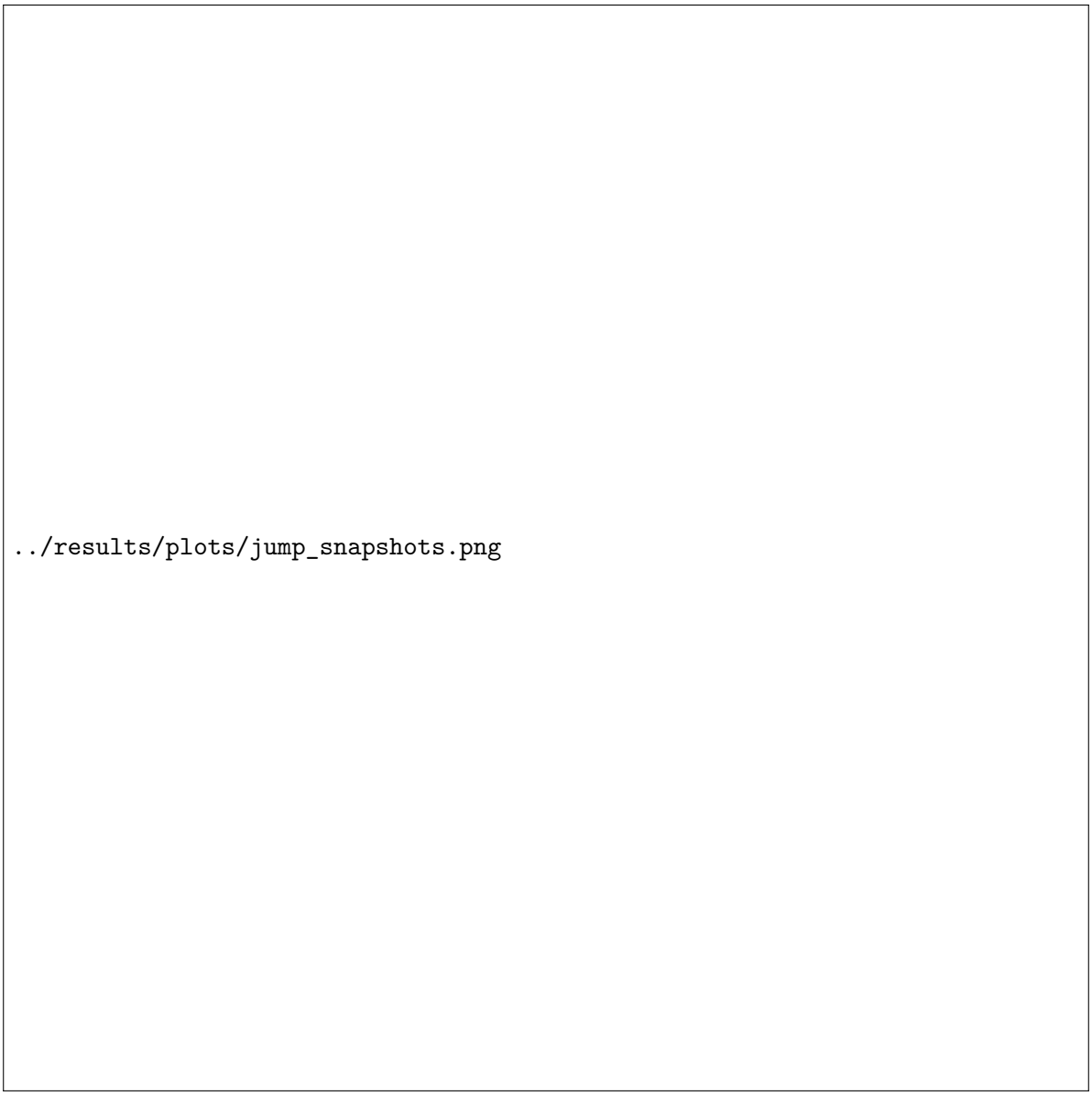

Figure 7: Jump snapshots showing initial detection windows (blue) and refined precise boundaries (yellow).

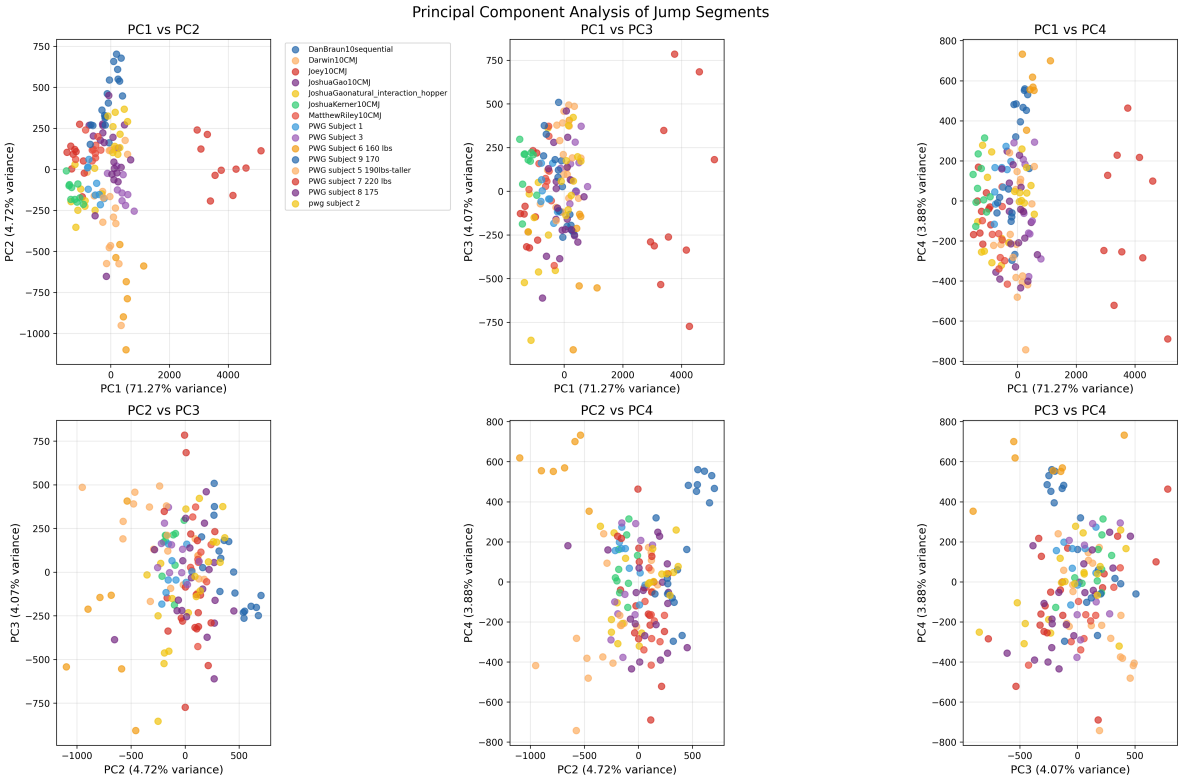

Figure 8: PCA of jump segments showing participant-specific clustering patterns.

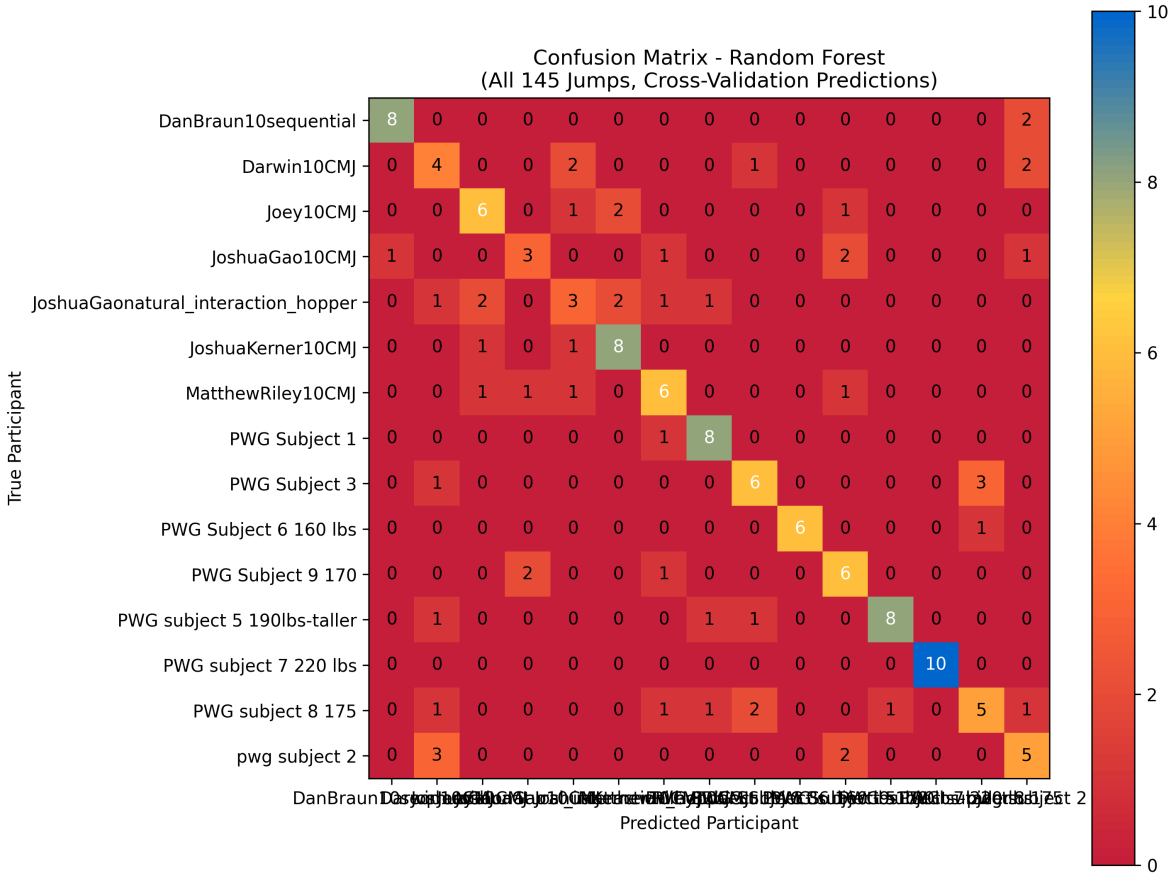

Figure 9: Confusion matrix for SVM classification of jumpers using PCA features.