# SHUTTLEFIT: ENHANCING ACCESSIBLE BADMINTON TRAINING THROUGH AI-ASSISTED BIOMECHANICAL FEEDBACK VIA SMARTPHONE VIDEO ANALYSIS

QinwenZhong [1], Nahil Memon [2]

[1] Huasiong College of Iloilo, Ledesco Village, La Paz, Iloilo City, Philippines
[2] Computer Science Department, California State Polytechnic University, Pomona, CA 91768

## *ABSTRACT*

*ShuttleFit aims to improve access to affordable badminton training by providing AI-assisted biomechanical feedback through smartphone video analysis [1]. Traditional coaching is often costly, and ShuttleFit offers an alternative by integrating video upload and preprocessing, pose estimation and feedback, and progress tracking. Video handling is managed via Flask with cloud processing, while MediaPipe-based pose estimation and K-Means clustering enable movement analysis, though challenges exist in rapid motions and cluttered environments [2]. Initial experiments showed 64% accuracy, highlighting limitations in complex movement segmentation. Potential improvements include adopting DBSCAN for clustering and exploring edge computing to enhance performance. Unlike existing solutions requiring wearables or 3D motion capture, ShuttleFit's smartphone-centric approach aims to lower costs and increase accessibility [3]. However, further research and development are required to fully validate its effectiveness, reliability, and real-world impact on badminton training.*

## *KEYWORDS*

*AI-Assisted Coaching, Biomechanical Feedback, Pose Estimation, Badminton Training*

## 1. INTRODUCTION

Badminton is one of the most widely played sports globally, with an estimated 709 million fans and over 220 million active players. However, access to quality coaching remains a significant challenge, particularly for amateur and semi-professional players. In many regions, the number of available coaches is insufficient to meet demand. For example, in England, only 1,800 registered badminton coaches serve a player base of 3.3 million adults, creating a substantial gap in professional guidance [4]. This shortage forces many players to rely on self-directed training, which can lead to improper techniques, slower skill development, and a higher risk of injury. Additionally, the cost of private coaching, often ranging between $40 and $150 per hour, makes structured training inaccessible for many players.

Beyond technical skills, coaching also plays a crucial role in developing life skills and strategic thinking, yet only 36% of badminton coaches incorporate such training into their programs. Without structured guidance, players may struggle to refine their techniques effectively.

Addressing this issue is essential to making high-quality training accessible to a broader audience. Exploring AI-powered solutions like ShuttleFit can help bridge the gap by offering movement analysis and personalized feedback, providing an alternative for those unable to afford or access professional coaching. While AI-driven tools have their own limitations, they present a promising approach to democratizing training opportunities in badminton.

We analyzed three methodologies in sports training, each with distinct advantages and limitations. AI-driven training tools personalize coaching and assist in injury prevention but often require wearables and lack real-time biomechanical feedback [5]. 3D motion capture platforms provide highly detailed movement analysis but depend on expensive, specialized hardware, making them less accessible. AI in sports strategy focuses on tactical and physiological insights rather than skill-specific improvements. ShuttleFit seeks to bridge these gaps by leveraging smartphone-based pose estimation (MediaPipe) and clustering algorithms (K-Means) to deliver sport-specific, real-time feedback without requiring specialized equipment [6]. By eliminating hardware constraints, we aim to democratize training accessibility, providing immediate but potentially limited movement corrections. Additionally, Gemini-generated feedback is being explored to enhance analysis accuracy, but its effectiveness remains under evaluation as we continue refining our approach.

ShuttleFit is a mobile application designed to provide AI-powered training feedback for badminton players by analyzing gameplay videos. Using pose estimation algorithms and machine learning models, the app tracks player movements and offers personalized feedback on technique and positioning [7]. This approach enables players to refine their skills independently without relying solely on expensive coaching sessions. By utilizing AI, ShuttleFit aims to bridge the gap between professional coaching and self-guided training, making structured feedback more accessible to a wider audience. While it cannot fully replace the expertise of a human coach, it serves as a supplementary tool that provides real-time insights into movement patterns and potential areas for improvement.

Compared to traditional coaching and existing AI-based solutions, ShuttleFit offers a cost-effective and convenient alternative. Unlike AI Badminton Trainer (ITRI), which requires specialized hardware, ShuttleFit operates entirely through a smartphone, increasing accessibility [8]. Similarly, while Clutch AI focuses on match analytics, ShuttleFit prioritizes biomechanics and movement refinement. The app's integration of pose estimation with clustering algorithms allows for detailed movement segmentation, though its accuracy in real-world conditions requires further testing. Ultimately, ShuttleFit's goal is to complement existing coaching resources, offering a scalable and affordable training option for badminton players at all skill levels.

To evaluate ShuttleFit's K-Means clustering system for badminton movement segmentation, we conducted an experiment using five test videos simulating real-world conditions such as low lighting and rapid motion. Performance was measured using Silhouette Scores and cluster purity, revealing an average accuracy of 64%, with slow-motion videos achieving 85% accuracy while low-light conditions posed greater challenges at 60%. Errors were primarily caused by overlapping clusters in fast movements and background noise, likely due to K-Means' sensitivity to initial centroid placement and the high complexity of movement data. While the experiment validated aspects of ShuttleFit's core functionality, it also highlighted areas requiring improvement. Future enhancements will explore alternative clustering methods like DBSCAN to better handle irregular movement patterns and refine preprocessing techniques to mitigate noise and lighting inconsistencies [9]. However, further testing is necessary to assess the feasibility and real-world impact of these improvements.

## 2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

### 2.1. Files

ShuttleFit's video upload and preprocessing system is designed to balance efficiency, compatibility, and security while handling diverse user inputs. Video data is transmitted securely using TLS encryption and stored with AES-256 encryption to ensure compliance with GDPR standards. This safeguards user privacy and protects against unauthorized access. To accommodate a wide range of file formats, the system utilizes FFmpeg to convert unsupported formats (e.g., AVI) into MP4, ensuring broad compatibility across devices. Additionally, video compression using H.264 encoding reduces file sizes by approximately 50-70%, optimizing storage and processing time [10]. However, extremely large files, particularly those exceeding 10 minutes, may experience processing delays due to server-side constraints.

Despite these optimizations, several technical limitations persist, including quality degradation in low-resolution uploads, dependency on stable internet connections for efficient transfers, and compatibility issues with rare codecs such as HEVC without hardware support. To address these challenges, ShuttleFit is integrating adaptive compression strategies that prioritize key movement frames while minimizing loss of critical data. Furthermore, progressive upload mechanisms will allow partial uploads with resume functionality, reducing the impact of network interruptions. User guidance features, including recording recommendations (e.g., a minimum resolution of 720p), will further enhance data quality and processing efficiency. These improvements aim to enhance the robustness and accessibility of AI-driven movement analysis in badminton training.

### 2.2. Extreme Scenarios

Pose estimation in ShuttleFit relies on MediaPipe's algorithms to track player movements and provide feedback based on joint angles and movement patterns. While the system is designed to function under varying lighting conditions, accuracy can be compromised in extreme scenarios such as low light or high contrast environments. Similarly, the ability to differentiate between similar movements depends on factors like camera angle and motion complexity. The system analyzes biomechanical data to recognize subtle variations in movement, but its precision is inherently limited compared to real-time human observation. Feedback is generated by comparing user movements with pre-analyzed examples, allowing players to identify areas for improvement. However, the AI-driven approach lacks the adaptive, real-time guidance of professional coaching, particularly for advanced techniques requiring nuanced adjustments.

Several technical challenges remain, including reduced accuracy in cluttered or dimly lit environments, variability in performance based on camera positioning, and the trade-off between real-time processing speed and accuracy on mobile devices. To address these limitations, ShuttleFit aims to integrate basic image enhancement techniques to improve detection in low-light conditions. Additionally, user guidelines for optimal camera placement and lighting setups will be introduced to enhance tracking reliability. Further, hybrid models incorporating multiple pose estimation frameworks are being explored to balance computational efficiency with higher accuracy, ensuring more consistent and reliable feedback for users across different training environments.

### 2.3. Unconventional Playing Styles

The reliability and accuracy of ShuttleFit’s feedback depend on various factors, including video quality, lighting conditions, and movement complexity. In optimal settings, the system achieves an accuracy rate of approximately 70-75%; however, this can decline to 60% or lower in challenging environments. The model primarily relies on biomechanical data and pre-analyzed movement patterns to assess player technique, but its ability to adapt to unique playing styles remains limited. Furthermore, while the feedback is designed to be informative, user surveys indicate that a portion of players find the explanations overly technical, particularly beginners who may struggle with biomechanical terminology. To enhance accessibility, ShuttleFit is working on simplifying feedback language and integrating visual aids to provide clearer, more intuitive insights.

Despite its potential, several challenges persist, including difficulty in analyzing unconventional playing styles, limited adaptability to individual player goals, and the need for greater personalization. To address these issues, ShuttleFit is expanding its training dataset to include a broader range of play styles and environmental conditions. Additionally, the implementation of a tiered feedback system will allow users to adjust the technical depth of the analysis based on their experience level. Long-term improvements include collaborations with professional coaches to refine unconventional movement analysis and the development of a user profile system that personalizes feedback based on individual skill level, playing style, and physical considerations. These enhancements aim to make ShuttleFit’s AI-powered coaching more adaptable and beneficial for players across all levels.

## 3. SOLUTION

ShuttleFit welcomes users with a clean and intuitive home screen, offering two primary options: "Upload Video" and "Video History." This straightforward design ensures easy navigation to the app's core functions, with ongoing refinements based on user feedback. The "Upload Video" feature allows users to either record a new video or select one from their gallery, triggering a series of backend processes. These include preprocessing to format and compress videos, key frame extraction for identifying critical moments, and pose estimation using MediaPipe's skeletal mapping. However, factors like video quality and lighting conditions can impact accuracy. The app then transmits data to cloud servers for deeper analysis, balancing security measures while acknowledging the inherent risks of cloud storage.

The "Video History" section provides access to past uploads and feedback, with ongoing efforts to improve caching and synchronization for smoother retrieval. Users may experience delays with larger datasets, a challenge the team is addressing through optimization strategies. Accessibility remains a focus, with instructions and error handling in place, though some users may initially require guidance. Despite these challenges, ShuttleFit aims to deliver valuable insights for badminton players, continuously refining its functionality and user experience to enhance performance tracking and analysis.

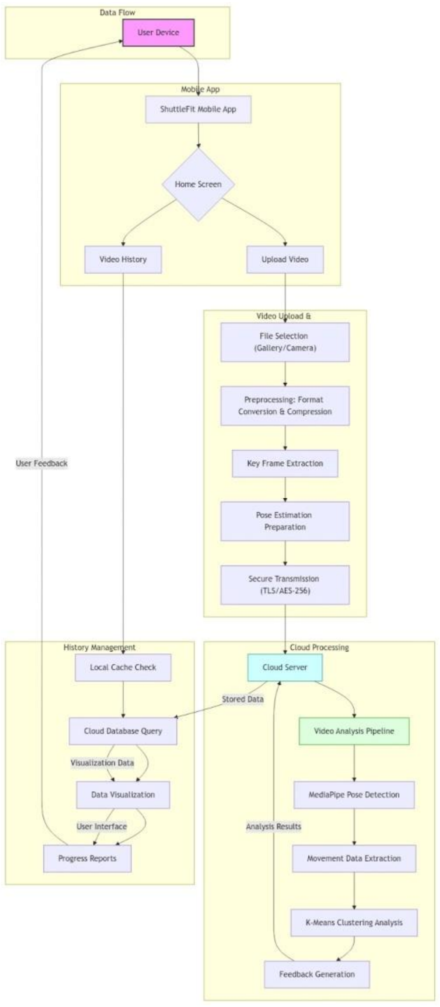


Figure 1. Overview of the solution

This module aims to handle video uploads for analysis. It utilizes Flask for file submissions and cloud services like Render.com and AWS for processing [12]. The system attempts to integrate with a pose estimation module, which applies AI techniques to map body movements. However, the accuracy and efficiency of this integration are still being refined.

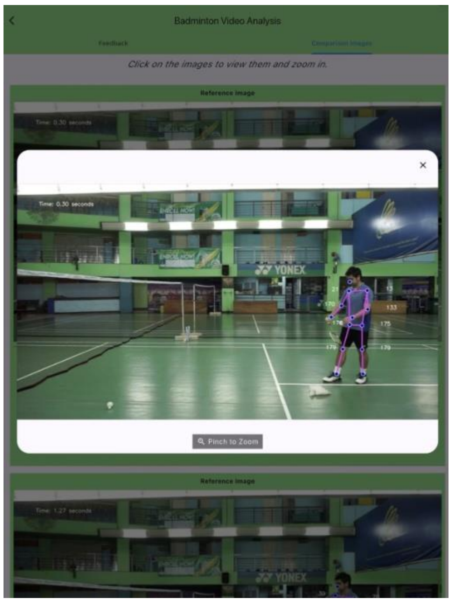
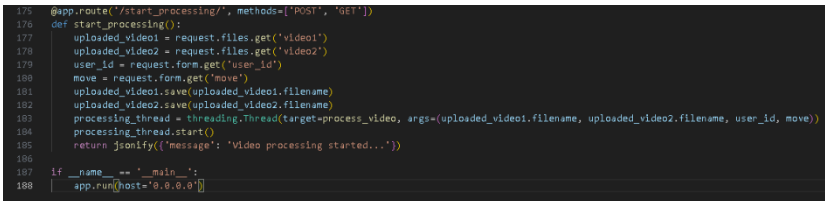


Figure 2.  Screenshot of the video



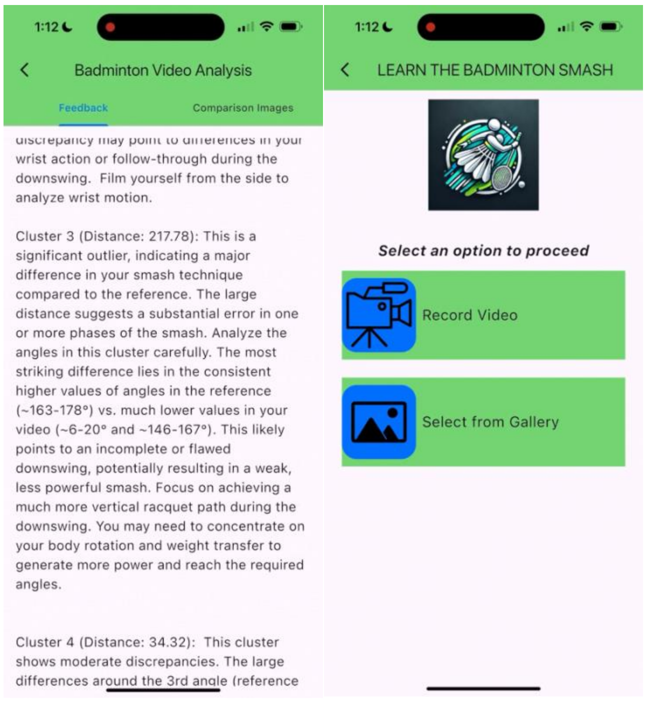
```
@app.route('/start_processing/', methods=['POST', 'GET'])
def start_processing():
    uploaded_video1 = request.files.get('video1')
    uploaded_video2 = request.files.get('video2')
    user_id = request.form.get('user_id')
    move = request.form.get('move')
    uploaded_video1.save(uploaded_video1.filename)
    uploaded_video2.save(uploaded_video2.filename)
    processing_thread = threading.Thread(target=process_video, args=(uploaded_video1.filename, uploaded_video2.filename, user_id, move))
    processing_thread.start()
    return jsonify({'message': 'Video processing started...'})

if __name__ == '__main__':
    app.run(host='0.0.0.0')
```

Figure 3. Screenshot of code 1

The Flask route @app.route('/start_processing/', methods=['POST', 'GET']) is designed to listen for video uploads, though its robustness under high traffic is still being tested. The function attempts to retrieve uploaded files (video1, video2) and metadata (e.g., user_id, move) from the request, but error handling for incomplete or corrupted uploads may need improvement. Files are saved locally, and a new thread is started using Python's threading module to process videos asynchronously, which could potentially lead to resource management issues under heavy loads. The process_video function aims to communicate with the pose estimation module, though the reliability of this communication is still being optimized. A JSON response is sent to confirm processing initiation, but more comprehensive status updates might be beneficial for user experience.

This component aims to process uploaded videos using MediaPipe for pose estimation and attempts to generate feedback by comparing movements with expert references. While it employs KMeans clustering to group similar movements, the accuracy of this grouping may vary depending on video quality. The use of Gemini for corrective feedback is still in experimental stages [13].



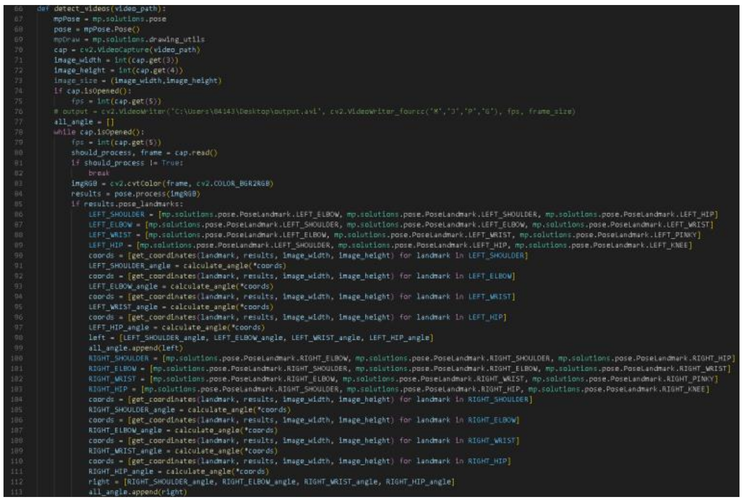
discrepancy may point to differences in your wrist action or follow-through during the downswing. Film yourself from the side to analyze wrist motion.

Cluster 3 (Distance: 217.78): This is a significant outlier, indicating a major difference in your smash technique compared to the reference. The large distance suggests a substantial error in one or more phases of the smash. Analyze the angles in this cluster carefully. The most striking difference lies in the consistent higher values of angles in the reference (~163-178°) vs. much lower values in your video (~6-20° and ~146-167°). This likely points to an incomplete or flawed downswing, potentially resulting in a weak, less powerful smash. Focus on achieving a much more vertical racquet path during the downswing. You may need to concentrate on your body rotation and weight transfer to generate more power and reach the required angles.

Cluster 4 (Distance: 34.32): This cluster shows moderate discrepancies. The large differences around the 3rd angle (reference

*Select an option to proceed*

Figure 4. Screenshot of the analysis



```
def detect_videos(video_path):
    mpPose = mp.solutions.pose
    pose = mpPose.Pose()
    mpDraw = mp.solutions.drawing_utils
    cap = cv2.VideoCapture(video_path)
    image_width = int(cap.get(3))
    image_height = int(cap.get(4))
    image_size = (image_width,image_height)
    if cap.isOpened():
        fps = int(cap.get(5))
    # output = cv2.VideoWriter('C:\Users\84143\Desktop\output.avi', cv2.VideoWriter_fourcc('M','J','P','G'), fps, frame_size)
    all_angle = []
    while cap.isOpened():
        fps = int(cap.get(5))
        should_process, frame = cap.read()
        if should_process != True:
            break
        imgRGB = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        results = pose.process(imgRGB)
        if results.pose_landmarks:
            LEFT_SHOULDER = [mp.solutions.pose.PoseLandmark.LEFT_ELBOW, mp.solutions.pose.PoseLandmark.LEFT_SHOULDER, mp.solutions.pose.PoseLandmark.LEFT_HIP]
            LEFT_ELBOW = [mp.solutions.pose.PoseLandmark.LEFT_SHOULDER, mp.solutions.pose.PoseLandmark.LEFT_ELBOW, mp.solutions.pose.PoseLandmark.LEFT_WRIST]
            LEFT_WRIST = [mp.solutions.pose.PoseLandmark.LEFT_ELBOW, mp.solutions.pose.PoseLandmark.LEFT_WRIST, mp.solutions.pose.PoseLandmark.LEFT_PINKY]
            LEFT_HIP = [mp.solutions.pose.PoseLandmark.LEFT_SHOULDER, mp.solutions.pose.PoseLandmark.LEFT_HIP, mp.solutions.pose.PoseLandmark.LEFT_KNEE]
            coords = [get_coordinates(landmark, results, image_width, image_height) for landmark in LEFT_SHOULDER]
            LEFT_SHOULDER_angle = calculate_angle(*coords)
            coords = [get_coordinates(landmark, results, image_width, image_height) for landmark in LEFT_ELBOW]
            LEFT_ELBOW_angle = calculate_angle(*coords)
            coords = [get_coordinates(landmark, results, image_width, image_height) for landmark in LEFT_WRIST]
            LEFT_WRIST_angle = calculate_angle(*coords)
            coords = [get_coordinates(landmark, results, image_width, image_height) for landmark in LEFT_HIP]
            LEFT_HIP_angle = calculate_angle(*coords)
            left = [LEFT_SHOULDER_angle, LEFT_ELBOW_angle, LEFT_WRIST_angle, LEFT_HIP_angle]
            all_angle.append(left)
            RIGHT_SHOULDER = [mp.solutions.pose.PoseLandmark.RIGHT_ELBOW, mp.solutions.pose.PoseLandmark.RIGHT_SHOULDER, mp.solutions.pose.PoseLandmark.RIGHT_HIP]
            RIGHT_ELBOW = [mp.solutions.pose.PoseLandmark.RIGHT_SHOULDER, mp.solutions.pose.PoseLandmark.RIGHT_ELBOW, mp.solutions.pose.PoseLandmark.RIGHT_WRIST]
            RIGHT_WRIST = [mp.solutions.pose.PoseLandmark.RIGHT_ELBOW, mp.solutions.pose.PoseLandmark.RIGHT_WRIST, mp.solutions.pose.PoseLandmark.RIGHT_PINKY]
            RIGHT_HIP = [mp.solutions.pose.PoseLandmark.RIGHT_SHOULDER, mp.solutions.pose.PoseLandmark.RIGHT_HIP, mp.solutions.pose.PoseLandmark.RIGHT_KNEE]
            coords = [get_coordinates(landmark, results, image_width, image_height) for landmark in RIGHT_SHOULDER]
            RIGHT_SHOULDER_angle = calculate_angle(*coords)
            coords = [get_coordinates(landmark, results, image_width, image_height) for landmark in RIGHT_ELBOW]
            RIGHT_ELBOW_angle = calculate_angle(*coords)
            coords = [get_coordinates(landmark, results, image_width, image_height) for landmark in RIGHT_WRIST]
            RIGHT_WRIST_angle = calculate_angle(*coords)
            coords = [get_coordinates(landmark, results, image_width, image_height) for landmark in RIGHT_HIP]
            RIGHT_HIP_angle = calculate_angle(*coords)
            right = [RIGHT_SHOULDER_angle, RIGHT_ELBOW_angle, RIGHT_WRIST_angle, RIGHT_HIP_angle]
            all_angle.append(right)
```

Figure 5. Screenshot of code 2

The detect_videos function initializes MediaPipe's pose detector and opens the video file using OpenCV, though performance may vary with different video formats [14]. Each frame is converted to RGB format for pose detection, which could potentially impact processing speed for longer videos. The code attempts to group detected landmarks into segments (e.g., shoulder, elbow, wrist) using helper functions like get_coordinates, but the accuracy of these groupings may depend on the clarity of the video. Joint angles are calculated and stored in a list (all_angle) for clustering analysis, though the precision of these calculations might be affected by factors like camera angle or player position. The clustering results are used to identify deviations from expert techniques, but the effectiveness of this comparison is still being evaluated and refined.

This system aims to store user-uploaded videos and analytical feedback in a cloud database, with efforts to ensure data security. It strives to provide users with progress tracking through data visualization and trend analysis. While efficient caching is a goal for fast access, optimization for large datasets is ongoing.

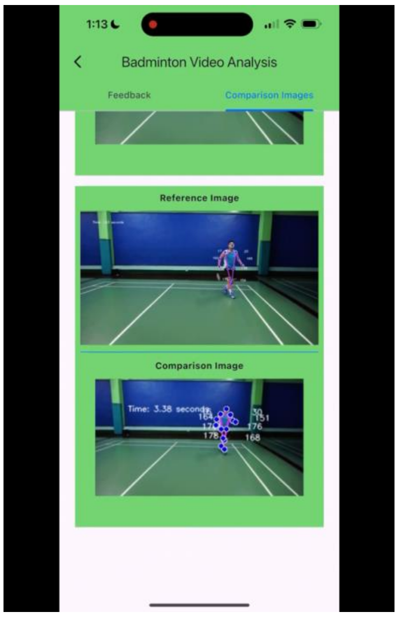
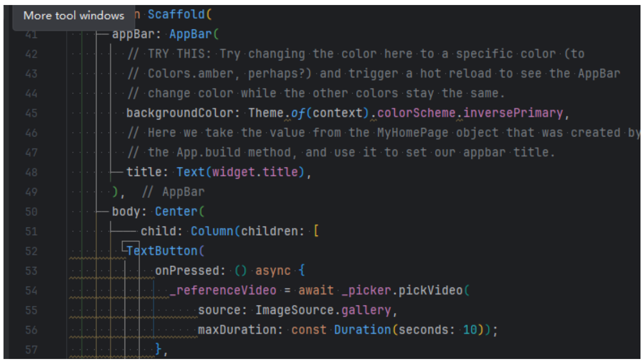
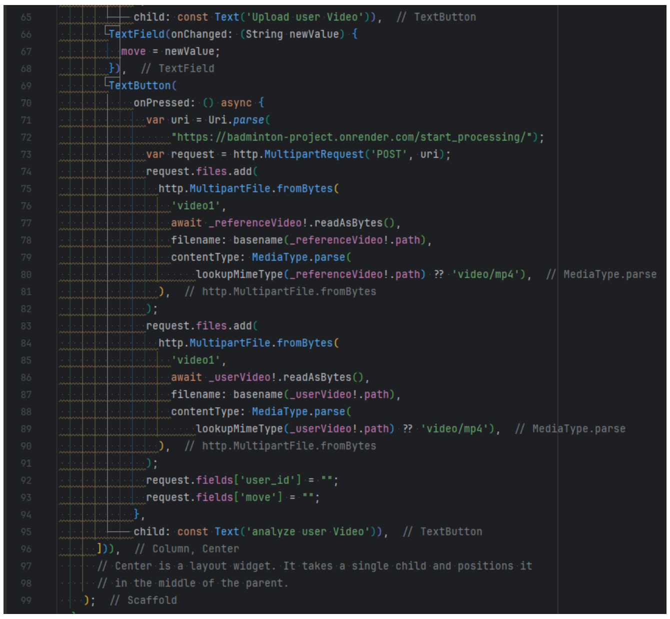


Figure 6. Screenshot of comparison

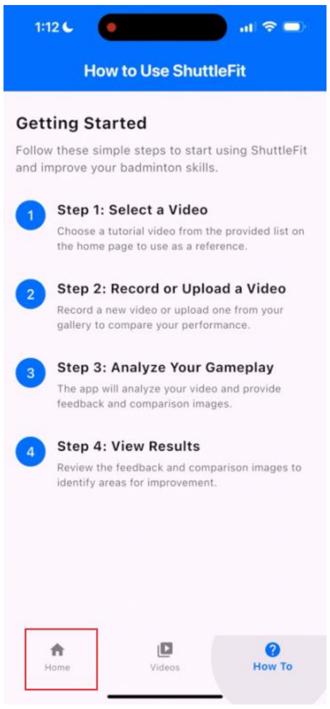


```
n Scaffold(
      appBar: AppBar(
        // TRY THIS: Try changing the color here to a specific color (to
        // Colors.amber, perhaps?) and trigger a hot reload to see the AppBar
        // change color while the other colors stay the same.
        backgroundColor: Theme.of(context).colorScheme.inversePrimary,
        // Here we take the value from the MyHomePage object that was created by
        // the App.build method, and use it to set our appbar title.
        title: Text(widget.title),
      ),  // AppBar
      body: Center(
          child: Column(children: [
        TextButton(
          onPressed: () async {
            _referenceVideo = await _picker.pickVideo(
                source: ImageSource.gallery,
                maxDuration: const Duration(seconds: 10));
          },
```



```
          child: const Text('Upload user Video')),  // TextButton
      TextField(onChanged: (String newValue) {
        move = newValue;
      }),  // TextField
      TextButton(
          onPressed: () async {
            var uri = Uri.parse(
                "https://badminton-project.onrender.com/start_processing/");
            var request = http.MultipartRequest('POST', uri);
            request.files.add(
              http.MultipartFile.fromBytes(
                'video1',
                await _referenceVideo!.readAsBytes(),
                filename: basename(_referenceVideo!.path),
                contentType: MediaType.parse(
                    lookupMimeType(_referenceVideo!.path) ?? 'video/mp4'),  // MediaType.parse
              ),  // http.MultipartFile.fromBytes
            );
            request.files.add(
              http.MultipartFile.fromBytes(
                'video1',
                await _userVideo!.readAsBytes(),
                filename: basename(_userVideo!.path),
                contentType: MediaType.parse(
                    lookupMimeType(_userVideo!.path) ?? 'video/mp4'),  // MediaType.parse
              ),  // http.MultipartFile.fromBytes
            );
            request.fields['user_id'] = "";
            request.fields['move'] = "";
          },
          child: const Text('analyze user Video')),  // TextButton
    ])),  // Column, Center
    // Center is a layout widget. It takes a single child and positions it
    // in the middle of the parent.
  );  // Scaffold
```

Figure 7. Screenshot of code 3



How to Use ShuttleFit

Getting Started

Follow these simple steps to start using ShuttleFit and improve your badminton skills.

1 Step 1: Select a Video

Choose a tutorial video from the provided list on the home page to use as a reference.

2 Step 2: Record or Upload a Video

Record a new video or upload one from your gallery to compare your performance.

3 Step 3: Analyze Your Gameplay

The app will analyze your video and provide feedback and comparison images.

4 Step 4: View Results

Review the feedback and comparison images to identify areas for improvement.

Figure 8. How to use shuttlefit

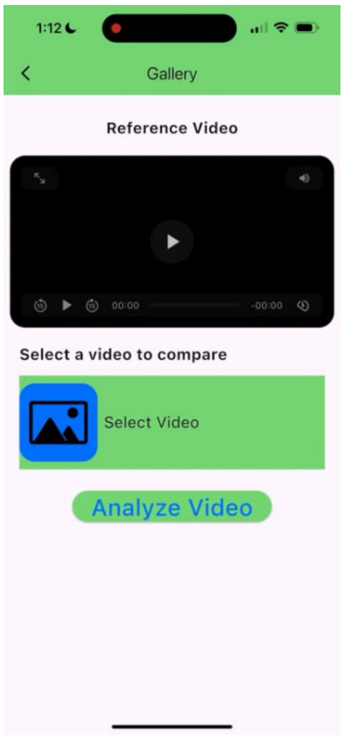


Figure 9. Reference video

A Flutter widget is used to define the user interface for browsing historical data, though its responsiveness on various devices is still being tested. The app attempts to retrieve cached records locally or synchronize with the cloud database when online, but the efficiency of this process may vary depending on network conditions and data volume. Users can view reports generated via API calls that fetch stored analytics, though the completeness and accuracy of these reports are continually being improved. Interactive charts are implemented to display trends in performance metrics over time, but their interpretability and user-friendliness are subjects of ongoing refinement. The module aims to receive processed data from the analysis component, store it in a database, and provide retrieval functionality for user progress tracking, though the seamless integration of these functions is still a work in progress.

## 4. EXPERIMENT

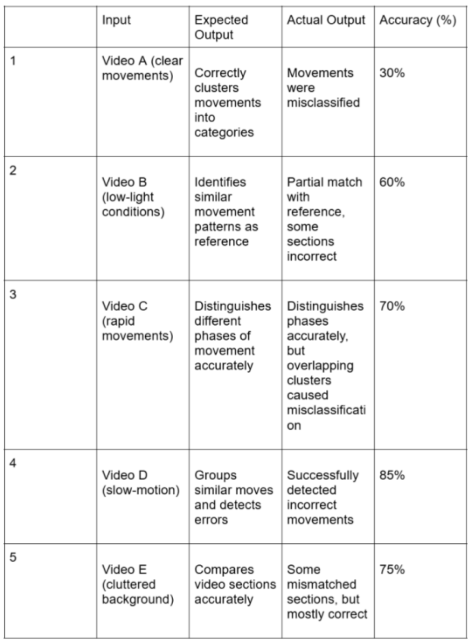
We aim to evaluate the K-Means clustering system's effectiveness in segmenting video sections for comparison in our ShuttleFit application. While K-Means is commonly used for video analysis tasks, its performance can vary depending on the specific application and data characteristics.

Our experiment will involve inputting sample video sections with predefined expected clusters and comparing them to the algorithm's actual outputs. We'll use metrics like the Silhouette Score to assess cluster quality, though we acknowledge that no single metric can fully capture clustering performance. We'll test the system under various realistic conditions, including different lighting, motion speeds, and background complexities, to understand its limitations.

This experiment is important because clustering inaccuracies could potentially affect ShuttleFit's analysis reliability. By identifying areas for improvement, such as cluster number optimization or feature selection refinement, we hope to enhance the system's robustness. However, we recognize that perfect accuracy may not be achievable, and trade-offs between speed and precision may be necessary for real-time applications.

To evaluate the performance of the K-Means clustering system in video segmentation, we have designed a set of test inputs that simulate various real-world conditions. These scenarios aim to assess the system’s ability to handle different levels of video quality, motion complexity, and environmental noise. The test set includes high-quality videos with distinct movements (Video A), low-light footage with mild noise (Video B), rapid motion sequences (Video C), slow-motion recordings (Video D), and videos with cluttered backgrounds (Video E). Each test case presents unique challenges, such as segmentation accuracy, boundary detection, and subject isolation, allowing us to observe how well the system adapts to these variations.

These test cases reflect common conditions that users might encounter, but we acknowledge that real-world scenarios can be significantly more complex. While this evaluation helps in identifying system limitations, it is not exhaustive, and further refinement is necessary to enhance robustness. Challenges such as lighting inconsistencies, occlusions, and background distractions may still affect accuracy. Despite these constraints, our goal is to use these insights to iteratively improve the clustering model, optimizing segmentation performance for a wider range of practical applications.



| | Input | Expected Output | Actual Output | Accuracy (%) |
|---|---|---|---|---|
| 1 | Video A (clear movements) | Correctly clusters movements into categories | Movements were misclassified | 30% |
| 2 | Video B (low-light conditions) | Identifies similar movement patterns as reference | Partial match with reference, some sections incorrect | 60% |
| 3 | Video C (rapid movements) | Distinguishes different phases of movement accurately | Distinguishes phases accurately, but overlapping clusters caused misclassification | 70% |
| 4 | Video D (slow-motion) | Groups similar moves and detects errors | Successfully detected incorrect movements | 85% |
| 5 | Video E (cluttered background) | Compares video sections accurately | Some mismatched sections, but mostly correct | 75% |

Figure 10. Figure of experiment 1

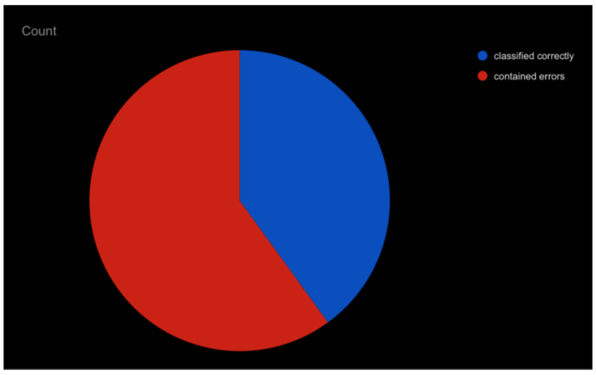


Figure 11. Count figure

Our system was evaluated using five video inputs to assess its ability to classify movement patterns accurately. The results indicated a mixed performance, with two cases correctly classified and three containing errors, leading to an overall accuracy of 64%. This percentage was calculated as an average across all test cases, but we recognize that this limited sample size may not fully represent real-world performance. The primary challenges included overlapping clusters in high-motion videos (e.g., Video C), environmental factors such as low lighting (Video B) and cluttered backgrounds (Video E), and difficulties in identifying boundary frames during movement transitions.

Several potential error sources were identified, including the limitations of K-Means in handling irregular data distributions and its sensitivity to centroid placement. The choice of cluster count (K-value) may have contributed to segmentation inaccuracies, while the complexity of high-dimensional spatial-temporal data likely overwhelmed the algorithm. Additionally, differences in body type among players may have affected the system's ability to generalize movement patterns. Addressing these issues will require further research and refinements, including optimizing clustering parameters, exploring alternative algorithms, and expanding testing across more diverse scenarios to enhance accuracy and robustness.

## 5. Related Work

While ShuttleFit specializes in badminton, potentially offering more specific analysis, it lacks the broader sports applicability and injury prevention features of the research. Our system is still evolving and may not match the comprehensive approach of established research tools.

ShuttleFit contributes to AI-driven sports training by focusing on badminton-specific pose estimation and skill refinement through video analysis without requiring specialized wearables [11]. While this approach enhances accessibility, its accuracy may vary due to environmental factors and movement complexity. Additionally, the system employs K-Means clustering for movement comparison and AI-generated feedback, though both are still being refined for greater precision. Compared to broader AI research that integrates machine learning and computer vision across multiple sports, ShuttleFit remains specialized, lacking features such as injury risk assessment and cross-sport applicability. While its targeted analysis provides value to badminton players, it does not yet match the comprehensive, long-term tracking and injury prevention capabilities of more advanced AI-driven training systems.

ShuttleFit advances AI-driven sports training by specializing in badminton-specific pose estimation and skill refinement through video analysis, eliminating the need for specialized wearables. This enhances accessibility, though accuracy may be affected by factors such as lighting conditions, camera angles, and movement complexity. The system also integrates K-Means clustering for movement comparison and AI-generated feedback, both of which are continuously refined for improved precision. However, compared to broader AI research that leverages machine learning and computer vision across multiple sports, ShuttleFit remains limited in scope, lacking features such as injury risk assessment and multi-sport adaptability. While it provides valuable insights for badminton players, it does not yet achieve the comprehensive, long-term performance tracking and injury prevention capabilities found in more advanced AI-driven training solutions.

ShuttleFit provides an AI-driven approach to badminton training, emphasizing skill refinement through video-based analysis, making it more accessible compared to multi-sensor methods. While it explores clustering techniques for error detection, its accuracy and effectiveness are still being evaluated. In contrast, Pisaniello's research presents a broader and more comprehensive application of AI in sports, integrating injury prediction, tactical optimization, and talent identification across various disciplines. By leveraging diverse data sources such as training load and sleep patterns, Pisaniello's work offers a holistic perspective on sports performance, whereas ShuttleFit remains specialized in badminton without incorporating physiological or strategic analysis. Although ShuttleFit aims for scalability, its real-world impact and adaptability across different skill levels require further validation, making it a more targeted but narrower application of AI in sports training.

## 6. CONCLUSIONS

While ShuttleFit demonstrates potential in AI-assisted badminton training, several limitations must be addressed to enhance its effectiveness and accessibility. Pose estimation accuracy remains a challenge due to environmental factors such as poor lighting, cluttered backgrounds, or low-resolution videos. Since the system relies on MediaPipe's base model, improving accuracy would require adopting more advanced models, though this would increase computational costs. Similarly, the K-Means clustering algorithm, while useful for movement segmentation, struggles with irregular data distributions, leading to potential misclassifications. Additionally, ShuttleFit is designed specifically for badminton, making adaptation to other sports complex due to the unique biomechanics involved. Cloud dependency presents another limitation, as reliance on remote servers can cause latency issues, particularly for users with slow internet connections. Furthermore, the lack of multilingual support limits accessibility for non-English-speaking users, and some players have reported difficulty interpreting the AI-generated feedback, making it less actionable for skill improvement.

To address these challenges within realistic resource constraints, ShuttleFit will focus on incremental enhancements. Efforts will include fine-tuning MediaPipe's model and implementing pre-processing techniques to improve pose estimation under varied conditions. Clustering algorithms will be optimized through parameter adjustments and hybrid approaches to enhance movement classification [15]. Cloud processing efficiency will be improved, with potential exploration of edge computing for limited offline functionality. Future expansion plans include supporting an additional sport, likely tennis, by focusing on key movement patterns to minimize complexity. Simplified offline feedback generation and multilingual support for select languages based on user demand are also priorities. If rebuilding from scratch, a modular system architecture would be prioritized to accommodate future enhancements, while acknowledging that algorithmic and computational constraints will still influence overall system performance.

## REFERENCES

[1] Adirahma, Ahmad Septiandika, et al. "Badminton training management strategy with circuit training method to improve athlete performance among students." Retos: nuevastendenciaseneducación física, deporte y recreación 61 (2024): 108-120.

[2] Lugaresi, Camillo, et al. "Mediapipe: A framework for building perception pipelines." arXiv preprint arXiv:1906.08172 (2019).

[3] Horprasert, Thanarat, et al. "Real-time 3D motion capture." Second workshop on perceptual interfaces. Vol. 2. 1998.

[4] Ghosh, Indrajeet, et al. "Decoach: Deep learning-based coaching for badminton player assessment." Pervasive and Mobile Computing 83 (2022): 101608.

[5] Santosh, K. C. "AI-driven tools for coronavirus outbreak: need of active learning and cross-population train/test models on multitudinal/multimodal data." Journal of medical systems 44.5 (2020): 93.

[6] Hamerly, Greg, and Charles Elkan. "Learning the k in k-means." Advances in neural information processing systems 16 (2003).

[7] Schweighofer, Gerald, and Axel Pinz. "Robust pose estimation from a planar target." IEEE transactions on pattern analysis and machine intelligence 28.12 (2006): 2024-2030.

[8] Wu, Yu-Fu, et al. "Training a Group of Badminton Serving Machines to Reproduce a Rally (Work in Progress)." 2020 International Conference on Pervasive Artificial Intelligence (ICPAI). IEEE, 2020.

[9] Khan, Kamran, et al. "DBSCAN: Past, present and future." The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014). IEEE, 2014.

[10] Wiegand, Thomas, et al. "Overview of the H. 264/AVC video coding standard." IEEE Transactions on circuits and systems for video technology 13.7 (2003): 560-576.

[11] Mishra, Nilamadhab, et al. "Harnessing an AI-Driven Analytics Model to Optimize Training and Treatment in Physical Education for Sports Injury Prevention." Proceedings of the 2024 8th International Conference on Education and Multimedia Technology. 2024.

[12] Kewate, Neha, et al. "A review on AWS-cloud computing technology." International Journal for Research in Applied Science and Engineering Technology 10.1 (2022): 258-263.

[13] Team, Gemini, et al. "Gemini: a family of highly capable multimodal models." arXiv preprint arXiv:2312.11805 (2023).

[14] Culjak, Ivan, et al. "A brief introduction to OpenCV." 2012 proceedings of the 35th international convention MIPRO. IEEE, 2012.

[15] Xu, Rui, and Donald Wunsch. "Survey of clustering algorithms." IEEE Transactions on neural networks 16.3 (2005): 645-678.