MATCHMOTION: AN AI-POWERED MOBILE APPLICATION FOR AUTOMATED BADMINTON VIDEO ANALYSIS USING YOLOV8 OBJECT DETECTION

Jinyuan Li [1], Jonathan Thamrun [2]

[1] Shenzhen College of International Education, No. 3 Antuoshan Sixth Road, Xiangmihu Street, Futian District, Shenzhen, Guangdong 518043
[2] University of California, Irvine, Irvine, CA 92697

ABSTRACT

Badminton is one of the world's most popular racket sports, yet accessible automated game analysis tools remain unavailable to recreational players. This paper presents MatchMotion, a cross-platform mobile application that enables users to upload badminton match videos for automated AI-powered analysis. The system employs three specialised YOLOv8 object detection models to identify players, shuttlecocks, and court boundaries within video frames, with a Python Flask backend hosted on AWS performing frame-by-frame inference and a Flutter frontend providing cross-platform mobile access. Custom algorithms analyse detection results to extract game statistics, including rally length, hit counts, shuttle landing positions, and rally winners. Experimental evaluations on ten BWF World Championship rally videos yielded a mean shuttle detection accuracy of 37.6% and a mean hit recognition accuracy of 81.7%, demonstrating that velocity-based hit detection effectively compensates for frame-level detection limitations. MatchMotion democratizes badminton video analysis by providing automated insights previously accessible only through expensive professional systems.

KEYWORDS

Badminton video analysis, YOLOv8 object detection, Flutter mobile application, Shuttlecock tracking

1. INTRODUCTION

Badminton is one of the fastest racket sports in the world, with shuttlecock speeds exceeding 400 km/h during professional smashes, making real-time analysis exceptionally challenging [1]. Despite being the second most played sport globally with over 220 million regular participants, the availability of accessible game analysis tools for recreational players remains severely limited [2]. Professional tournaments such as the BWF World Championships employ sophisticated Hawk-Eye tracking systems for line-calling and broadcast analysis, yet these technologies are prohibitively expensive and inaccessible to the vast majority of amateur players [3]. Current badminton coaching relies predominantly on subjective observation and manual video review, which is both time-consuming and prone to human error [4]. Coaches and players who wish to analyze match footage must manually scrub through recordings, count shots, identify patterns, and assess positioning without automated assistance. This process can require hours for a single match and often yields incomplete results. The absence of affordable automated analysis tools creates a significant barrier to skill improvement for the estimated 200 million casual players worldwide

David C. Wyld et al. (Eds): CSITEC, NLPI, CMCA, NECOM, DMML, SP, ADCO – 2026
pp. 151-163, 2026. CS & IT - CSCP 2026 DOI: 10.5121/csit.2026.160813

who lack access to professional coaching infrastructure. Furthermore, existing sports analysis platforms such as Hudl and Dartfish are designed primarily for team sports like football and basketball, offering minimal functionality tailored to badminton-specific metrics such as shuttle trajectory, rally length, and court coverage [5]. The few badminton-specific solutions that do exist, such as the TIVEE system, require pre-processed data including exact player positions and shuttle trajectories, limiting their applicability to controlled research environments [6]. This gap between professional-grade analysis and consumer accessibility represents a significant unmet need in sports technology [7].

Three existing methodologies were compared against MatchMotion. The TIVEE system by Chu et al. provides immersive 3D VR-based visual analytics for professional coaches but requires pre-processed input data and specialized VR hardware, making it inaccessible for recreational use with raw video. TrackNetV3 by Chen and Wang achieve state-of-the-art shuttlecock tracking accuracy through specialized temporal architectures and trajectory rectification but operates as a standalone tracking model without integrated game analysis, hit detection, or mobile delivery. The system by Hsu, Yu, and Cheng fuses shuttlecock tracking with hit detection from monocular camera footage but remains a research tool without mobile deployment or user-facing features. All three solutions address only subsets of the automated analysis pipeline. MatchMotion improves upon these works by providing a complete, automated end-to-end system combining detection, analysis, and mobile delivery in a single accessible platform.

The proposed solution is MatchMotion, a full-stack cross-platform mobile application that enables users to upload badminton match videos for automated AI-powered analysis and annotation. The application employs three specialized YOLOv8 object detection models to identify players, shuttlecocks, and court boundaries within video frames [8]. The system is built using the Flutter framework for cross-platform mobile deployment on both iOS and Android, a Python Flask server hosted on Amazon Web Services for computational processing, and Google Firebase for cloud-based authentication, database storage, and file hosting [9][10]. When a user uploads a video, the Flask server processes it frame-by-frame at 512x512 resolution, applying the three YOLOv8 models to detect and track game objects. Custom Python algorithms then analyze the positional data to extract meaningful statistics including rally length, number of hits per player, shuttle landing position relative to court boundaries, and winner determination. The annotated video with bounding boxes around detected objects is uploaded back to Firebase for the user to view. MatchMotion differentiates itself from existing solutions by providing a fully automated end-to-end pipeline that processes raw user-uploaded videos without requiring manual preprocessing or specialized equipment [11]. Unlike research prototypes that target controlled laboratory environments, MatchMotion is designed for practical recreational use, accepting standard smartphone-quality video as input [12]. By combining state-of-the-art object detection with an intuitive mobile interface and cloud-based processing, MatchMotion democratizes badminton game analysis, making performance insights previously available only through expensive professional systems accessible to a broad base of recreational and amateur players [13].

Two experiments were conducted to evaluate MatchMotion's core analysis capabilities using ten rally videos from the BWF World Championships 2025. The first experiment assessed frame-level shuttle detection accuracy using a custom OpenCV validation tool. The mean accuracy was 37.6% across all ten videos, with performance ranging from 28.0% to 50.0%. Motion blur, small object size, and visual confusion with similarly colored court elements were identified as the primary detection failure sources. The second experiment evaluated velocity-based hit recognition accuracy against manually established ground truth, achieving a substantially higher mean accuracy of 81.7% with a range of 69% to 93%. The hit recognition algorithm demonstrated effective compensation for frame-level detection failures through its velocity-change-based approach. False positive hits outnumbered false negatives, indicating an over-detection bias attributable to false shuttle

detections on stationary objects. Together, these experiments confirm that while shuttle detection requires improvement, the overall analysis pipeline provides actionable game statistics for recreational players.

2. CHALLENGES

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

2.1. Improving Shuttlecock Detection for Fast Small Objects

Shuttlecock detection presents a fundamental technical challenge due to the extreme speed and small physical size of the shuttle during flight. At frame rates of 30 fps, the shuttlecock frequently appears as a motion-blurred streak or becomes entirely invisible between frames, making consistent detection difficult. The YOLOv8 model achieves a mean detection accuracy of only 37.6% across test videos. To address this, one could train on higher-resolution datasets that include motion-blurred examples, employ temporal context from adjacent frames to interpolate shuttle positions, or utilize specialized small-object detection architectures such as attention-based feature pyramid networks.

2.2. Reducing Latency in Multi-Model Video Processing

Processing video frame-by-frame through three separate YOLOv8 models on a remote server introduces significant computational overhead and latency. Each uploaded video must be transmitted to the AWS-hosted Flask server, analyzed frame-by-frame, annotated, and re-uploaded to Firebase Cloud Storage. This pipeline can take several minutes even for short clips. To mitigate this, one could implement model quantization to reduce inference time, use batched frame processing rather than sequential analysis, deploy optimized models using TensorFlow Lite for potential on-device inference, or leverage GPU-accelerated cloud instances to parallelize the three detection models across available hardware.

2.3. Improving Player Identification and Tracking

Distinguishing between players and correctly attributing shots to the appropriate player presents a non-trivial algorithmic challenge, particularly during net exchanges when players occupy similar court positions. The system determines player identity by calculating the court midpoint using detected court boundaries and classifying each detected player as “top” or “bottom” based on vertical position. However, this heuristic could fail when players approach the midpoint or when court boundary detection is inaccurate. One could implement player re-identification using appearance-based features such as jersey color classification or employ a tracking algorithm that maintains persistent player identities across consecutive frames.

3. SOLUTION

The MatchMotion application employs three-tier architecture comprising a Flutter mobile frontend, a Python Flask backend server, and Google Firebase cloud services. The Flutter frontend provides an intuitive interface for video upload, library management, and analysis visualization across both iOS and Android platforms. The Flask server, hosted on AWS, acts as computational middleware. When a user uploads a video for analysis, the Flutter application transmits the file to the Flask server, which orchestrates the entire analysis pipeline. The server runs three YOLOv8 models sequentially on each frame: a player detection model with a confidence threshold of 0.75, a shuttle detection model with a threshold of 0.1, and a court boundary detection model with a threshold of

0.3 executed only on the first frame since the court remains static. Detection results are processed by custom analysis algorithms that compute game statistics and generate annotated video output. Google Firebase provides three critical backend services: Firebase Authentication for secure user login and registration, Cloud Firestore as a NoSQL document database for storing user profiles and video metadata, and Cloud Storage for hosting both original and analyzed video files. The Flask server interacts with Firebase through its REST API and the Firebase Admin Python SDK, ensuring that all data access is authenticated. This architecture enables computationally intensive YOLOv8 inference to execute on a capable server rather than on the mobile device, making the application accessible to users with lower-end smartphones while maintaining analysis quality.

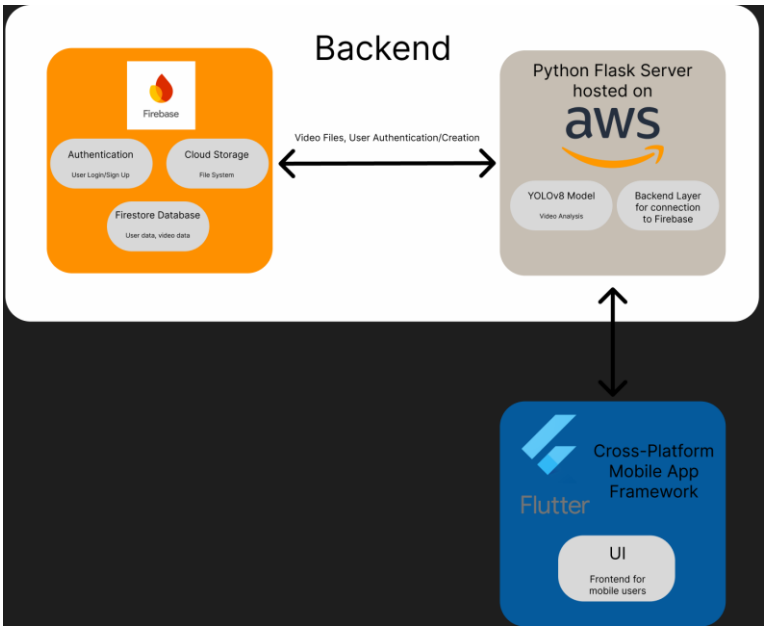

Figure 1. Architecture of the MatchMotion automated badminton video analysis system

The YOLOv8 video analysis component is the core machine learning engine of MatchMotion. It processes each video frame through three specialized detection models trained on datasets sourced from the Roboflow platform. This component implements object detection, overlap resolution, velocity-based hit detection, and player position classification using OpenCV for frame extraction and manipulation.

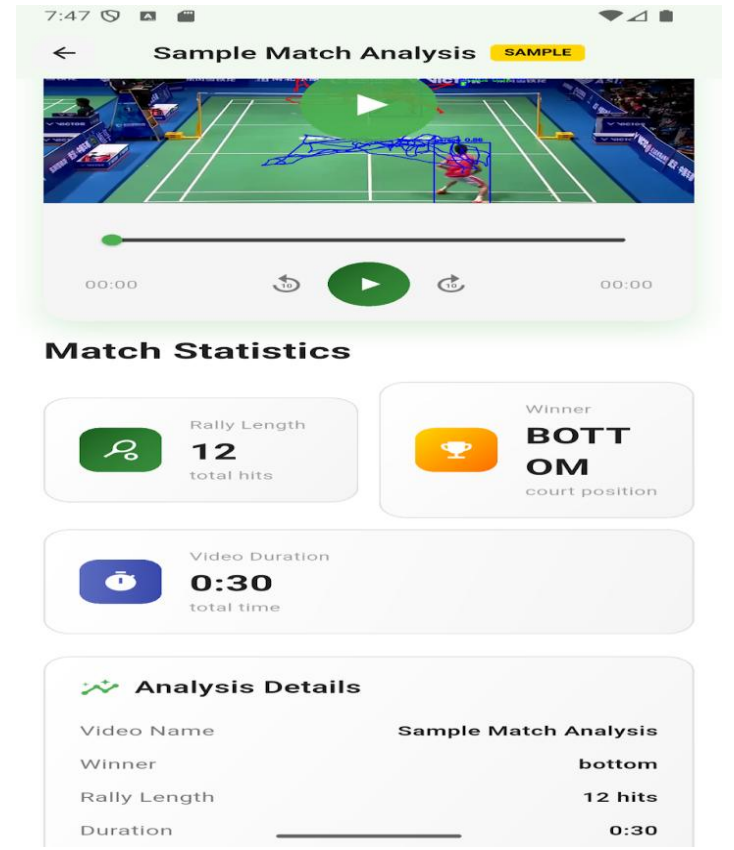

Figure 2. Interface of the MatchMotion mobile application showing analyzed badminton rally visualization

```python
def analyze_video(video_path):
    while True:
        ret, frame = stream.read()
        if ret:
            frames += 1
            print(f"progress:{((frames/framestotal)*100):.1f}%")
            # Run Inference
            results_player = model(frame, conf=0.75, verbose=False)
            results_shuttle = modelshuttle(frame, conf=0.1, verbose=False)
            if frames == 1:
                results_court = modelcourt(frame, conf=0.3, verbose=False)[0]
                court_bounds = results.court
                court_frame = results_court.plot()
                cv2.imwrite("court.png", court_frame)
            detections = []
            for result in [results_player[0], results_shuttle[0]]:
                boxes = result.boxes.xyxy.cpu().numpy()
                confs = result.boxes.conf.cpu().numpy()
                classes = result.boxes.cls.cpu().numpy().astype(int)
                names = result.names
                for (x1, y1, x2, y2), conf, cls in zip(
                    boxes, confs, classes
                ):
                    name = names[cls]
                    if name == "shuttle" and conf < 0.11:
                        name = "shuttle"
                    detections.append((x1, y1, x2, y2, conf, name))
```

Figure 3. Core video analysis pipeline implementing YOLOv8-based shuttle and player detection

The analyze_video function processes each frame of the uploaded video through the YOLOv8 detection pipeline. For every frame, the function executes player detection with a confidence threshold of 0.75 to ensure high-precision localization, and shuttle detection with a lower threshold of 0.1 to maximize recall on the small and fast-moving shuttlecock. Court boundary detection runs only on the first frame since the court position remains static throughout the video. The function extracts bounding box coordinates, confidence scores, and class labels from each model's output using NumPy array operations. All detections are aggregated into a unified list for post-processing, which includes overlap resolution to handle duplicate shuttle detections. When multiple shuttle bounding boxes overlap by more than 50%, only the detection with the higher confidence score is retained. This noise reduction step is critical for downstream hit detection accuracy, as false shuttle detections can trigger spurious hit events in the velocity-based analysis algorithm.

The Firebase integration layer manages all cloud-based data operations for MatchMotion, including user authentication, video storage, and metadata persistence. The system uses Firebase Authentication for secure email and password-based user registration and login, preventing unauthorized access to user-specific video data.

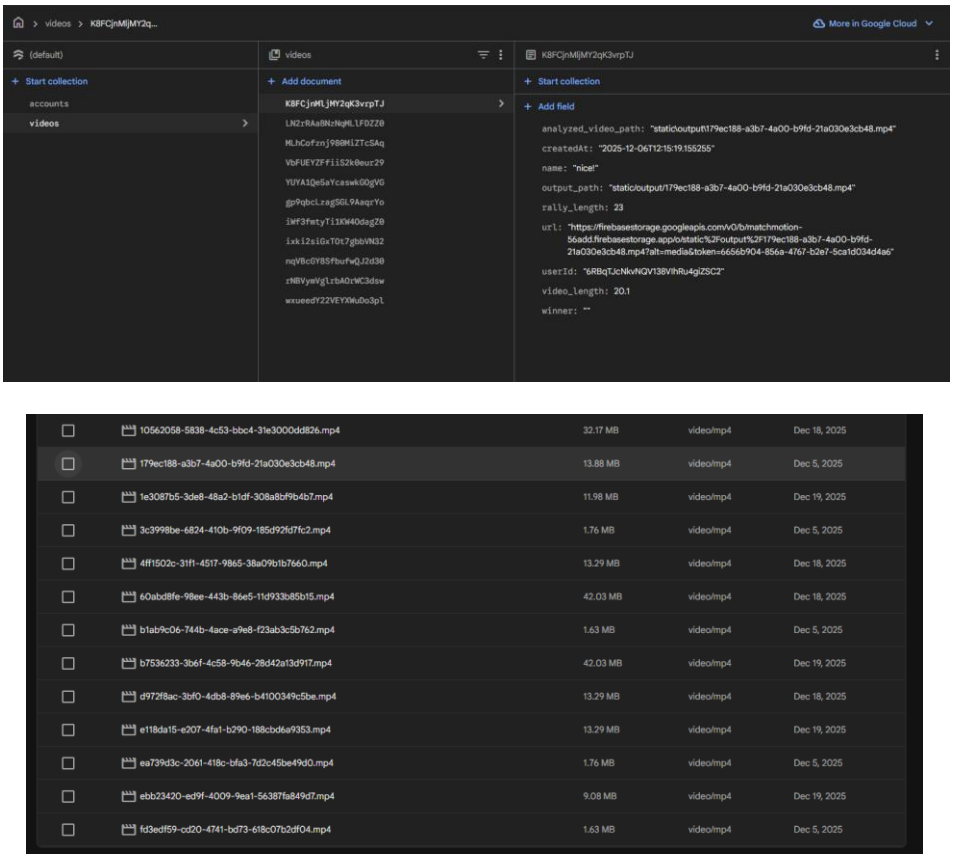

Figure 4. Mobile interface displaying analyzed rally statistics and shuttle trajectory

```
def register_user(self, email: str, password: str):
    try:
        user = auth.create_user(email=email, password=password)
        return {"uid": user.uid, "email": user.email}
    except Exception as e:
        raise ValueError(f"Error creating user: {e}")

def login_user(self, email: str, password: str):
    url =
f"https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?key={
self.api_key}"
    payload = {"email": email, "password": password, "returnSecureToken":
True}
    r = requests.post(url, json=payload)
    if r.status_code == 200:
        return r.json()
    else:
        raise ValueError(f"Login failed: {r.json()}")

def verify_user(self, id_token: str):
    try:
        decoded_token = auth.verify_id_token(id_token)
        return decoded_token
    except Exception as e:
        raise ValueError(f"Invalid authentication token: {e}")
```

Figure 5. Firebase authentication and token verification workflow within the backend server

The Firebase authentication code implements three core functions: user registration via the Firebase Admin SDK's create_user method, user login through Firebase's REST Identity Toolkit API, and token verification using verify_id_token to decode and validate encrypted authentication tokens. The login_user method sends credentials to Google's Identity Toolkit endpoint and returns a secure token upon successful authentication. The verify_user method decodes this token on the Flask server to extract the user's unique identifier, ensuring that all subsequent database and storage operations are scoped to the authenticated user. This architecture prevents direct client access to Firebase, routing all requests through the Flask server as an additional layer of security.

The Python Flask server serves as the computational bridge between the Flutter mobile client and the Firebase backend, exposing RESTful API endpoints for video upload, user authentication, and data retrieval. Flask was selected for its lightweight design and ease of integration with the YOLOv8 Python inference pipeline.

Figure 6. Example output video frame with detected players, shuttlecock, and court boundaries

```python
@app.route("/upload", methods=["GET", "POST"])
def upload_video():
    if request.method == "POST":
        if "file" not in request.files:
            return "No file part"
        file = request.files["file"]
        if file.filename == "":
            return "No selected file"
        if file:
            video_id = str(uuid.uuid4())
            filepath = os.path.join(app.config["UPLOAD_FOLDER"], video_id)
            file.save(filepath)
            # Run YOLO
            results = analyze_video(filepath)
            # Attach uid and name from app
            results["userId"] = (data_manager.verify_user(
                request.form.get("userId")))["uid"]
            results["name"] = request.form.get("name")
            results["createdAt"] = datetime.now().isoformat()
            response = data_manager.upload_to_storage(results["output_path"])
            results["url"] = response["url"]
            data_manager.create_document("videos", "", results)
            return jsonify({
                "message": "Upload successful",
                "video_id": video_id
            }), 200
```

Figure 7. Server-side REST API endpoint responsible for video upload and analysis orchestration

The /upload endpoint orchestrates the complete video analysis pipeline. Upon receiving a POST request containing a video file, the server generates a unique identifier using UUID, saves the file locally, and invokes analyze_video to run YOLOv8 inference. After analysis completes, the server verifies the user's authentication token, attaches metadata including the user ID, video name, and creation timestamp, and uploads the annotated output video to Firebase Cloud Storage. The resulting download URL and all analysis metadata have persisted as a Firestore document in the "videos" collection, linking the analyzed video to the authenticated user. This design ensures that video processing, authentication verification, storage upload, and database operations occur within a single request lifecycle.

4. EXPERIMENT

4.1. Experiment 1

This experiment evaluates the frame-by-frame accuracy of the YOLOv8 shuttle detection model across ten badminton rally videos sourced from professional BWF World Championship footage. Ten video clips were obtained from the BWF TV YouTube channel, each recorded at 1920x1080 resolution at 30 frames per second using Open Broadcaster Software. Each clip contains exactly one rally from the BWF World Championships 2025 match between Shi Yu Qi (CHN) and Kunlavut Vitidsarn (THA) during the third set. A custom Python validation tool built with OpenCV was developed to establish ground truth: the tool displays each frame sequentially, allowing a human reviewer to navigate with arrow keys and mark frames where shuttle detection is incorrect by pressing the spacebar. A successful detection requires exactly one shuttle bounding box correctly positioned on the actual shuttlecock, with no false positives when the shuttle is not visible.

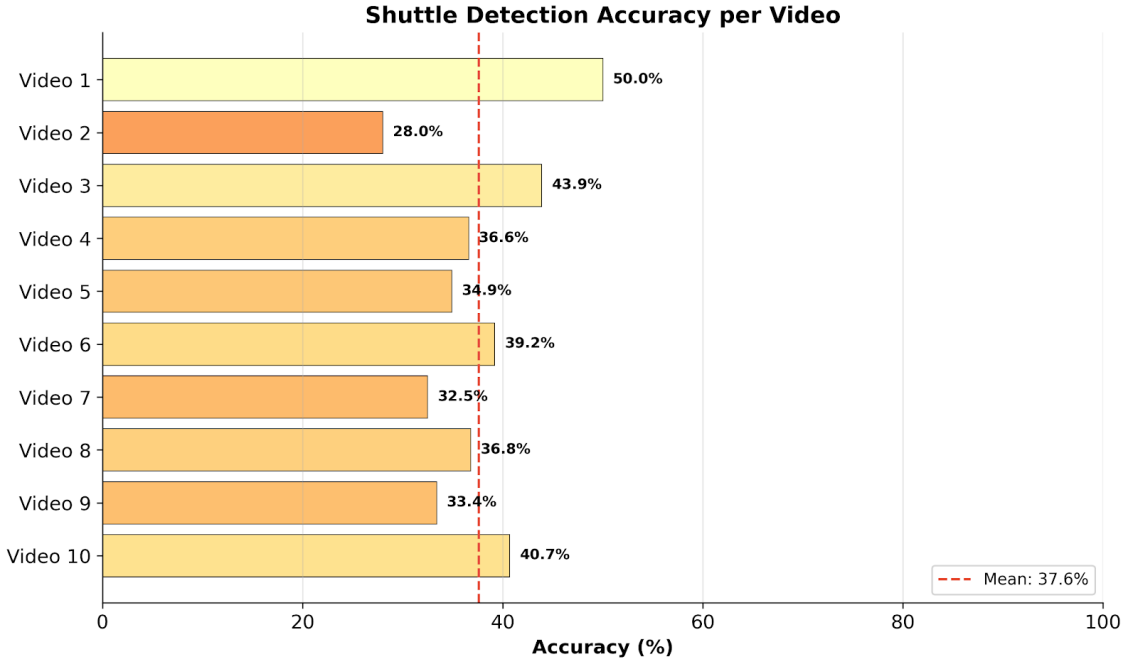

Figure 8. Shuttle detection accuracy across ten test videos. The mean accuracy is 37.6%, with Video 1 achieving the highest accuracy (50.0%) and Video 2 the lowest (28.0%)

Across the ten test videos, the YOLOv8 shuttle detection model achieved a mean frame-level accuracy of 37.6%, with individual video accuracies ranging from 28.0% (Video 2) to 50.0% (Video 1). The standard deviation of 6.1% indicates moderate consistency across different rally

conditions. Analysis reveals that shorter rallies did not consistently produce higher accuracy than longer rallies, suggesting that rally duration is not the primary factor affecting detection performance. The primary sources of detection failure include motion blur during high-speed shuttle movements, the shuttle's small pixel footprint relative to the full frame, and visual confusion with similarly-colored stationary objects such as court line markings and sponsor logos. Video 1 achieved the highest accuracy likely due to slower shuttle speeds during a net exchange, where the shuttle was more frequently visible and stationary. These results indicate that shuttle detection remains the principal bottleneck in the analysis pipeline and that model improvements would yield the greatest overall system improvement.

4.2. Experiment 2

This experiment evaluates the accuracy of the velocity-based hit recognition algorithm, which depends on both shuttle detection quality and the correctness of the hit detection logic.

The same ten rally videos were used for hit recognition evaluation. Ground truth was established by manually counting the number of shuttle hits by each player throughout each rally. The custom validation tool was then used to identify two categories of errors: false marked hits (the algorithm registers a hit when none occurred) and false no-hits (the algorithm fails to detect an actual hit). Hit recognition accuracy was calculated as the percentage of correctly identified hit events relative to the ground truth count, accounting for both false positives and false negatives.

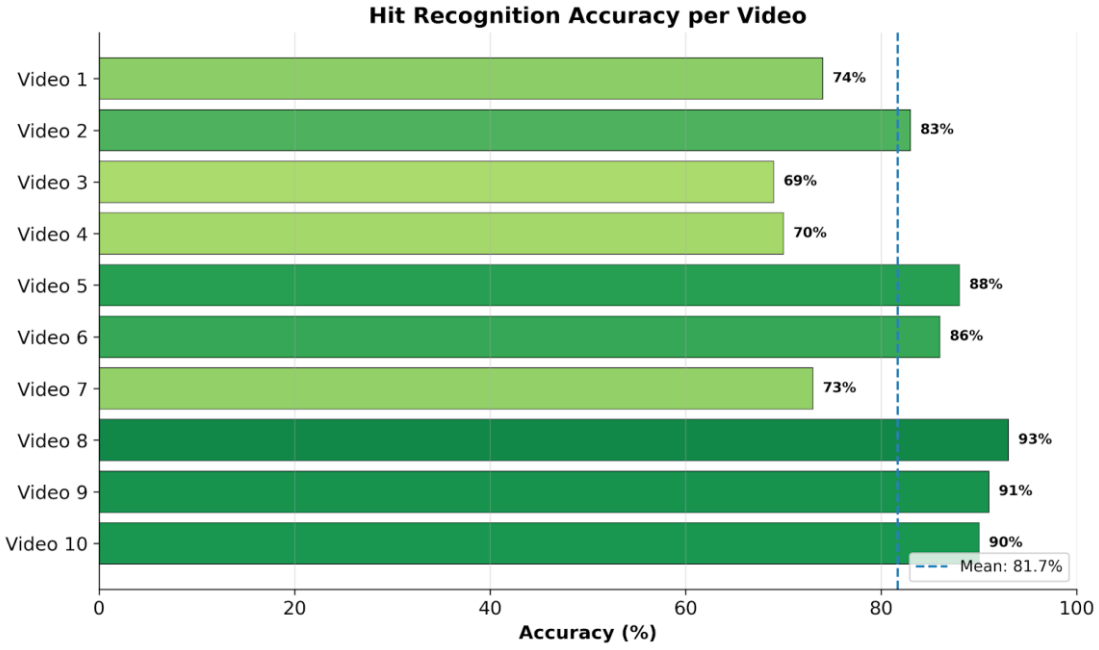

Figure 9. Hit recognition accuracy across ten test videos. The mean accuracy is 81.7%, substantially higher than shuttle detection accuracy

The hit recognition algorithm achieved a mean accuracy of 81.7% across all ten videos, with individual accuracies ranging from 69% (Video 3) to 93% (Video 8). This substantially outperforms the 37.6% shuttle detection accuracy, demonstrating that the velocity-based algorithm compensates effectively for frame-level detection failures. The algorithm detects hits by monitoring shuttle velocity changes across consecutive frames: when the shuttle's velocity exceeds a high percentile of recent velocity history, a hit is registered. False marked hits (mean: 4.6 per

video) were more frequent than false no-hits (mean: 1.4 per video), indicating a tendency toward over-detection. This bias stems primarily from false shuttle detections on stationary white objects, which generate artificial velocity spikes. Videos 8, 9, and 10 achieved the highest accuracies (93%, 91%, 90%), suggesting that the algorithm performs most reliably in certain rally configurations.

5. RELATED WORK

The TIVEE system developed by Chu et al. provides a three-dimensional virtual reality-based immersive visual analytics platform for badminton match analysis [6]. TIVEE offers rich visualization capabilities including 3D court reconstruction and player trajectory visualization, targeting professional coaches and analysts. However, TIVEE requires pre-processed input data including exact player positions, shuttle trajectories, and shot classifications, making it unsuitable for processing raw user-uploaded videos. Additionally, the VR interface limits accessibility to users with specialized hardware. MatchMotion addresses these limitations by providing a fully automated end-to-end pipeline that processes raw video footage without manual preprocessing, delivering results through a standard mobile interface accessible on any smartphone.

TrackNetV3, developed by Chen and Wang, represents the state-of-the-art in shuttlecock tracking using a specialized deep learning architecture with augmentation techniques and trajectory rectification [14]. TrackNetV3 achieves significantly higher tracking accuracy than general-purpose object detectors by leveraging temporal information across multiple frames and specialized heatmap-based detection. However, TrackNetV3 is a standalone tracking model without an integrated application framework for end-to-end video analysis. It does not provide game statistics, hit detection, or player attribution capabilities. MatchMotion could potentially integrate TrackNetV3 as a replacement for its YOLOv8 shuttle detection model to improve the 37.6% detection accuracy while retaining its comprehensive analysis pipeline and mobile delivery platform.

Hsu, Yu, and Cheng developed a badminton game analysis system that fuses shuttlecock tracking with hit detection from monocular camera footage [4]. Their approach uses a combination of TrackNet for shuttle tracking and rule-based hit detection, achieving strong performance on controlled broadcast datasets. However, their system is designed as a research tool for post-hoc analysis rather than a consumer-facing product. It lacks mobile deployment, user authentication, cloud storage, and the ability for end users to upload and analyze their own videos. MatchMotion improves upon this work by wrapping similar analysis capabilities within a complete mobile application stack that handles the full workflow from video upload to results visualization.

6. CONCLUSIONS

Several limitations warrant future attention. First, the shuttle detection accuracy of 37.6% remains the primary bottleneck, directly impacting downstream hit recognition reliability. Integrating specialized tracking architectures such as TrackNetV3 or implementing temporal context networks that leverage information from adjacent frames could substantially improve detection performance. Second, the current system processes videos entirely on the server, requiring stable internet connectivity and introducing latency. Deploying optimized models using TensorFlow Lite or ONNX Runtime for on-device inference could enable offline processing. Third, the player attribution algorithm relies on a simple court midpoint heuristic that fails during net exchanges or when court detection is inaccurate. Incorporating appearance-based player re-identification using embedding networks would improve attribution robustness. Fourth, the system currently supports only single-rally analysis from broadcast-quality video. Extending support to full-match processing

from amateur camera angles would broaden utility. Finally, expanding the training datasets with more diverse court conditions and lighting environments would improve model generalization.

This research demonstrates that combining YOLOv8 object detection with a cloud-based mobile application architecture enables automated badminton video analysis accessible to recreational players. Despite shuttle detection limitations, the velocity-based hit recognition algorithm achieves 81.7% accuracy, providing meaningful game insights previously available only through expensive professional systems or manual coaching analysis.

REFERENCES

[1] Ooi, Cheong Hwa, et al. "Physiological characteristics of elite and sub-elite badminton players." Journal of sports sciences 27.14 (2009): 1591-1599.
[2] Vaughan, Richard. "International Sports Federations Voting System: A Case Study of the Badminton World Federation." International Journal of Racket Sports Science 6.2 (2024).
[3] Owens, N. E. I. L., C. Harris, and C. Stennett. "Hawk-eye tennis system." 2003 international conference on visual information engineering VIE 2003. IET, 2003.
[4] Hsu, Yi-Hua, Chih-Chang Yu, and Hsu-Yung Cheng. "Enhancing badminton game analysis: an approach to shot refinement via a fusion of shuttlecock tracking and hit detection from monocular camera." Sensors 24.13(2024): 4372.
[5] Naik, Banoth Thulasya, Mohammad Farukh Hashmi, and Neeraj Dhanraj Bokde. "A comprehensive review of computer vision in sports: Open issues, future trends and research directions." Applied Sciences 12.9 (2022): 4429.
[6] Chu, Xiangtong, et al. "TIVEE: Visual exploration and explanation of badminton tactics in immersive visualizations." IEEE Transactions on Visualization and Computer Graphics 28.1 (2021): 118-128.
[7] Thomas, Graham, et al. "Computer vision for sports: Current applications and research topics." Computer Vision and Image Understanding 159 (2017): 3-18.
[8] Kukartsev, V. V., et al. "Deep learning for object detection in images development and evaluation of the yolov8 model using ultralytics and roboflow libraries." Computer Science On-line Conference. Cham: Springer Nature Switzerland, 2024.
[9] Suri, Bhawna, et al. "Cross-platform empirical analysis of mobile application development frameworks: Kotlin, react native and flutter." Proceedings of the 4th International Conference on Information Management & Machine Intelligence. 2022.
[10] Khawas, Chunnu, and Pritam Shah. "Application of firebase in android app development-a study." International Journal of Computer Applications 179.46 (2018): 49-53.
[11] Chu, Wei-Ta, and Samuel Situmeang. "Badminton video analysis based on spatiotemporal and stroke features." Proceedings of the 2017 ACM on international conference on multimedia retrieval. 2017.
[12] Menon, Akshay, et al. "A machine learning framework for shuttlecock tracking and player service fault detection." International Conference on Deep Learning Theory and Applications. Cham: Springer Nature Switzerland, 2023.
[13] Diwan, Tausif, G. Anirudh, and Jitendra V. Tembhurne. "Object detection using YOLO: challenges, architectural successors, datasets and applications." multimedia Tools and Applications 82.6 (2023): 9243-9275.
[14] Chen, Yu-Jou, and Yu-Shuen Wang. "Tracknetv3: Enhancing shuttlecock tracking with augmentations and trajectory rectification." Proceedings of the 5th ACM International Conference on Multimedia in Asia. 2023.
[15] Sun, Nien-En, et al. "Tracknetv2: Efficient shuttlecock tracking network." 2020 International Conference on Pervasive Artificial Intelligence (ICPAI). IEEE, 2020.
[16] Cao, Zhiguang, et al. "Detecting the shuttlecock for a badminton robot: A YOLO based approach." Expert Systems with Applications 164 (2021): 113833.
[17] Liu, Hongshan, et al. "Automated player identification and indexing using two-stage deep learning network." Scientific Reports 13.1 (2023): 10036.
[18] Zaidi, Syed Sahil Abbas, et al. "A survey of modern deep learning based object detection models." Digital Signal Processing 126 (2022): 103514.
[19] Wang, Tianyi, and Tongyan Li. "Deep Learning-Based Football Player Detection in Videos." Computational Intelligence and Neuroscience 2022.1 (2022): 3540642.

[20] Goh, Guo Liang, et al. "Automated service height fault detection using computer vision and machine learning for badminton matches." Sensors 23.24 (2023): 9759.

©2026 By AIRCC Publishing Corporation. This article is published under the Creative Commons Attribution (CC BY) license.