IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, November 2026
Impact Factor: 8.2

SmartRef: A Low-Cost Embedded Vision-Based Referee System for Real-Time Badminton Officiating Using Edge AI

Rohith K J[1], Mohamed Aaftaab M R[2], Rayhan Hameed[3], Amrita Kiran S[4]
[1,2,3,4] Department of Electronics and Communication Engineering (Advanced Communication Technology)
Chennai Institute of Technology, Chennai, India
rohithkj.act2024@citchennai.net

Abstract : *Human officiating in fast racket sports is limited by visibility, reaction time and bias, while automated line-calling and video-review systems such as Hawk-Eye depend on costly multi-camera installations and centralised processing that grassroots tournaments cannot afford. This paper presents SmartRef, a low-cost, portable, vision-based referee system that does all of its processing on a Raspberry Pi 4 Model B (8 GB) without any internet connection. Live video from a network camera (an IP camera or a smartphone streaming through DroidCam) or a pre-recorded match video is processed with OpenCV. A lightweight YOLOv5n/YOLOv8n detector locates the shuttlecock and the players, SORT/DeepSORT keeps their identities across frames, and a geometric rule checker compares their positions with a calibrated court boundary to flag ball-out and net-related events. Each violation triggers a GPIO-driven buzzer and LED, an overlay on the video frame and an entry in a Flask-based dashboard that offers a live view, a scoreboard, a timestamped event log and replay for human verification. In controlled indoor and outdoor tests, SmartRef correctly classified 46 of 50 ball in/out cases (92%) with a processing time of 50 to 80 ms per frame, worked stably up to 10 m from the camera and gave buzzer and LED alerts with 100% reliability. It was faster and more consistent than human officiating for ball in/out calls, and optimised processing with sleep cycles kept its power use low enough for portable set-ups. Accuracy dropped under occlusion and in low light. The results show that edge-based vision assistance for officiating is practical on low-cost commodity hardware for amateur and college-level badminton.*

Keywords: Embedded Vision, Raspberry Pi, Edge AI, YOLO, Object Tracking, Sports Officiating, Badminton, Real-Time Systems.

I. INTRODUCTION

Fair and consistent officiating is central to competitive sport. In fast racket sports such as badminton, the shuttlecock reaches very high speeds and changes direction within fractions of a second, so line judges and umpires have to decide under limited visibility and severe time pressure. A wrong call can change the outcome of a rally and, in the end, of a whole match.

Technology-assisted officiating has addressed this problem at the elite level. Ball-tracking line-calling systems such as Hawk-Eye reconstruct ball trajectories from several synchronised high-speed cameras [1], and video-review schemes allow officials to re-examine contested incidents. Hawk-Eye was first adapted for badminton in 2014, and research instant review systems for the sport still place many cameras around the court [2]. These systems are built around dedicated multi-camera installations, centralised processing and specialist operators, which puts them far beyond the budgets of school, college and local-club tournaments. Where such events are officiated at all, they still rely completely on human judgment.

Copyright to IJARSCT
www.ijarsct.co.in
DOI: 10.48175/IJARSCT-41001

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, November 2026
Impact Factor: 8.2

Recent progress in lightweight deep-learning detectors, notably the YOLO family [3], [4], [5], and in real-time multi-object tracking [6], [7] makes it realistic to run vision-based decision support on inexpensive single-board computers. Computer vision has already been applied to a wide range of sports-analysis tasks [8], [9], including shuttlecock and table-tennis ball tracking [10], [11] and the detection of service height faults in badminton [12]. What is still missing is a complete, standalone and affordable system that goes from camera input to an immediate, human-verifiable officiating decision on an edge device, without needing cloud services or an internet connection.

The work was guided by three research questions. The first was how computer vision and AI can be used to automate real-time officiating. The second was whether a low-cost, portable system can detect rule violations reliably, and the third was how edge-based processing can make such a system accessible to grassroots sport. Based on these questions, five objectives were set, namely real-time decision-making with AI models on a Raspberry Pi 4B, automatic detection of events such as ball in/out and net touches, a reduction in human officiating errors, a low-cost and portable design for grassroots and college tournaments, and a modular structure that can be extended to other sports and to extra cameras or sensors later.

This paper presents SmartRef, an embedded vision-based referee system implemented on a Raspberry Pi 4 Model B (8 GB RAM). SmartRef analyses a live network-camera stream or a recorded match video, detects the shuttlecock and the players, tracks them across frames and compares their positions with a calibrated court model to detect ball-out and net-related violations. Decisions are given instantly through a buzzer and LED, through an overlay on the video and through a local web dashboard that also supports replay-based checking by a human referee. Fig. 1 gives a pictorial overview of the idea. The main contributions of the work are as follows.

1) A fully standalone, offline architecture that brings video capture, detection, tracking, rule checking, alerting and a web dashboard together on a single Raspberry Pi 4B.

2) A calibration-based geometric rule checker for ball in/out and net events that does not depend on the detector used and can be adapted to other sports.

3) A multi-channel feedback and replay interface that keeps a human referee in the decision loop.

4) An experimental evaluation in indoor and outdoor conditions that reports detection accuracy, per-frame latency, operating range, alert reliability, failure cases and a comparison with human officiating.

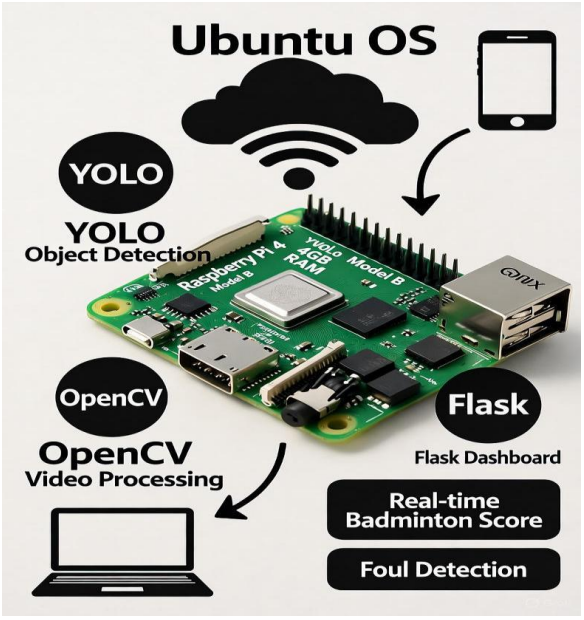

Fig. 1. Pictorial overview of SmartRef.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, November 2026
Impact Factor: 8.2

The rest of the paper is organised as follows. Section II reviews related work, Section III describes the system architecture, Section IV explains the methodology and the implementation, Section V describes the experimental set-up, Section VI presents and discusses the results, and Section VII concludes the paper and outlines future work.

II. RELATED WORK

A. Commercial and Research Officiating Systems

Hawk-Eye is the best-known automatic line-calling system. It estimates the path of the ball from several calibrated cameras and reports whether the ball landed in or out [1]. Video-review procedures in other sports, such as the video assistant referee in football, add a human review of recorded footage to the on-field decision. For badminton, Kopania et al. built an instant review system with 15 cameras placed around the court, used it at the BWF World Senior Championships and compared its calls with the decisions of the linesmen [2]. Goh et al. used 2 cameras, a workstation and a YOLOv5 detector to check service height faults automatically, and benchmarked their system against a motion capture system [12]. Zhao et al. developed a public AI referee assistance system on the Harmony OS platform for college students, who often play without sideline referees [13].

All of these systems give useful accuracy, but they need many cameras, a workstation-class computer or a dedicated platform. Sensor-based systems that place infrared or ultrasonic sensors on the court are cheaper, but they need changes to the court itself and only cover a few lines. Table I compares these approaches with SmartRef.

TABLE I: Qualitative comparison of officiating approaches

Approach	Sensing	Processing / decision	Relative cost
Hawk-Eye [1]	Multiple high-speed cameras	Dedicated central servers, automatic	High
Video review	Broadcast or multiple cameras	Operations room, human review	High
Instant review system [2]	15 cameras around the court	Dedicated computer with operator check	High
Service fault detection [12]	2 cameras	Workstation with YOLOv5, automatic	Moderate
Sensor-based	IR or ultrasonic sensors on court	Microcontroller, automatic	Low to moderate
SmartRef (proposed)	Single network camera	Raspberry Pi 4B (edge), automatic alert with human verification	Low

B. Computer Vision in Sports

Thomas et al. surveyed computer-vision applications in sport, including player and ball tracking, event detection and tactical analysis, and pointed out that the small size and high speed of the ball remain a persistent difficulty [8]. Naik et al. reviewed detection, tracking and trajectory prediction across sports such as soccer, cricket and badminton, and also discussed GPU workstations and embedded platforms for sports vision [9]. TrackNet tracks small, fast objects such as the badminton shuttlecock and the tennis ball with a deep heat-map network [10], and TTNet performs real-time temporal and spatial analysis of table-tennis video, including ball detection and event spotting [11]. These works show the value of vision-based event detection, but most of them target offline analysis or GPU-class hardware.

C. Real-Time Detection and Tracking

YOLO reformulated detection as a single-pass regression problem and made real-time inference possible [3]. The nano variants of YOLOv5 and YOLOv8 [5], [4] give up some accuracy in exchange for a very small footprint, which suits CPU-based edge devices. For tracking, SORT combines a Kalman-filter motion model with Hungarian assignment to

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, November 2026
Impact Factor: 8.2

link detections across frames at very high speed [6], and DeepSORT adds an appearance descriptor to reduce identity switches [7].

D. Edge Computing on Single-Board Computers

The Raspberry Pi 4 Model B offers a quad-core Arm Cortex-A72 CPU, up to 8 GB of RAM, GPIO pins and network connectivity on a low-cost, low-power board [14]. These properties make it attractive for self-contained monitoring and decision systems, as long as the vision pipeline is kept light. SmartRef builds on this combination to give a complete officiating aid for resource-constrained settings.

III. SYSTEM ARCHITECTURE

A. Hardware

SmartRef is built around a Raspberry Pi 4 Model B with 8 GB RAM running Ubuntu. The board was chosen because it combines a quad-core processor, USB-C power, 2 micro-HDMI ports, USB 2.0 and USB 3.0 ports, Gigabit Ethernet and GPIO pins on one small board [14]. Video is supplied by a network camera, which can be an IP camera or a smartphone running DroidCam that exposes the phone camera as an IP stream over Wi-Fi. A pre-recorded MP4 file stored on the SD card or on an external drive can also be used. A buzzer and an LED are connected to GPIO pins 18 and 23 (BCM numbering) of the Pi to give instant audio and visual feedback. A laptop is used only as a display and control terminal through SSH and VNC/XRDP, and all of the computation takes place on the Pi.

B. Software Stack

The software is written in Python. OpenCV [15] handles frame acquisition, pre-processing, geometry tests and overlays, the Ultralytics implementation of YOLO [4] performs object detection, SORT/DeepSORT [6], [7] provides tracking, RPi.GPIO drives the alert outputs, and Flask [16] serves a browser-based dashboard written in HTML, CSS and JavaScript. TensorFlow Lite [17] was used while testing model performance on the Pi. Table II lists the full configuration, and Fig. 2 shows the overall architecture.

TABLE II: Hardware and software configuration

Component	Specification
Processing board	Raspberry Pi 4 Model B, 8 GB RAM
Operating system	Ubuntu
Programming language	Python
Video input	IP camera or smartphone (DroidCam) over Wi-Fi, or an uploaded MP4 video
Detector	YOLOv5n / YOLOv8n (nano models)
Tracker	SORT / DeepSORT
Vision library	OpenCV (Python)
Model testing	TensorFlow Lite on the Raspberry Pi 4B
Alert outputs	Buzzer (GPIO 18), LED (GPIO 23)
Dashboard	Flask back end with an HTML, CSS and JavaScript front end
Remote access	SSH, VNC/XRDP over the local network

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, November 2026
Impact Factor: 8.2

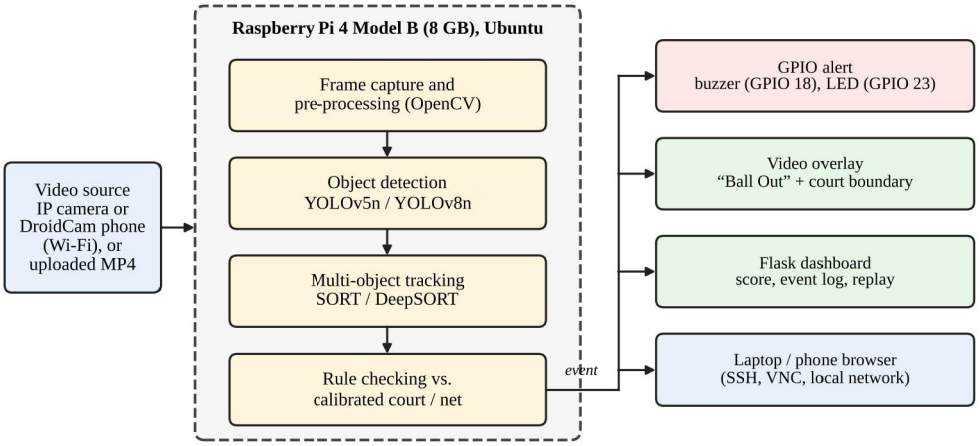

Fig. 2. Overall architecture of the SmartRef system.

C. Networking and Deployment

To allow quick set-up at different venues, the Pi is configured with static IP profiles for several Wi-Fi networks, so that the Pi, the camera and the monitoring device can join the same local network wherever the system is used. Fig. 3 shows the network configuration file, in which 4 profiles are defined for a home Wi-Fi network, a mobile hotspot, the college Wi-Fi and the campus network. Each profile fixes the IP address, the router address and the DNS servers of the Pi, so the dashboard can always be reached at a known address. No internet access is needed during operation, which means the system also works in venues that have no connectivity at all.

```
md@mohamed-desktop: ~
 GNU nano 6.2
interface wlan0
static ip_address=192.168.1.50/24
static routers=192.168.1.1
static domain_name_servers=8.8.8.8 8.8.4.4
# 1  Home Wi-Fi
interface wlan0
ssid "Airtel_moha_9786"
static ip_address=192.168.1.50/24
static routers=192.168.1.1
static domain_name_servers=8.8.8.8 8.8.4.4

# 2  Mobile Hotspot
interface wlan0
ssid "No Internet!"
static ip_address=192.168.43.50/24
static routers=192.168.43.1
static domain_name_servers=8.8.8.8 8.8.4.4

# 3  College Wi-Fi
interface wlan0
ssid "Aaftaab 77"
static ip_address=192.168.10.50/24
static routers=192.168.10.1
static domain_name_servers=8.8.8.8 8.8.4.4

# 4  Other Network
interface wlan0
ssid "CIT-Campus"
static ip_address=192.168.0.50/24
static routers=192.168.0.1
static domain_name_servers=8.8.8.8 8.8.4.4
```

Fig. 3. Static IP profiles configured on the Raspberry Pi for the network module.

D. Working Flow

Fig. 4 shows how the modules work together. The smartphone running DroidCam sends a live video feed over Wi-Fi, and the Raspberry Pi is reached over SSH. The network module keeps every device on the same network, while the AI

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, November 2026
Impact Factor: 8.2

processing module uses YOLO for fast object detection and OpenCV for real-time video processing. Its output goes to the dashboard interface, which feeds a replay buffer, a live display screen and a storage module. The replay buffer and the live display drive the monitor output, and the storage module is used to display the scores, the fouls and the replays.

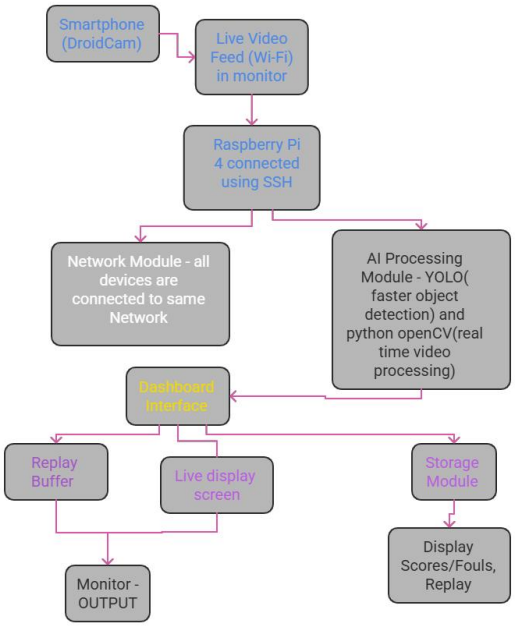

Fig. 4. Working flow of SmartRef from video capture to output.

IV. METHODOLOGY

A. Video Acquisition and Pre-Processing

Frames are read with the VideoCapture interface of OpenCV, either from the network stream of the camera (RTSP or HTTP) or from a locally stored video file. Each frame is resized and, where needed, denoised before it is passed to the detector. This keeps the input compatible with the model and limits the computational load on the Pi.

B. Object Detection

A nano-sized YOLO model (YOLOv5n or YOLOv8n) is used to detect the shuttlecock and the players in every frame. Nano variants were chosen because they can run on the CPU of the Pi within the real-time budget. Badminton video clips were used to train and evaluate the detection models. For each detection, the model outputs a bounding box (x_1, y_1, x_2, y_2), a confidence score and a class label, and the centre of the box is used as the object position c, given by

$$c = \left(\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2}\right) \qquad (1)$$

C. Multi-Object Tracking

Detections are passed to SORT/DeepSORT, which gives persistent identifiers to the shuttlecock and to each player across frames [6], [7]. Frame-to-frame association keeps the object identities stable and lets the system cope with short missed detections, which are common for a small, fast-moving shuttlecock.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, November 2026
Impact Factor: 8.2

D. Court Calibration and Rule Checking

During set-up, the court boundary is defined as a polygon B in image coordinates (for example, from the 4 court corners), and the net is defined as a separate region N. The rules are then checked geometrically. A ball-out event is raised when the centre c of a tracked shuttlecock lies outside B, which is tested with the point-in-polygon function of OpenCV. Net-related violations are checked in the same way by comparing the positions of the tracked objects with the calibrated net region. Because the rule checker works only on object positions, it does not depend on the particular detector and can be set up again for other courts and sports by changing the calibrated regions and rules. The processing loop is summarised in Algorithm 1.

Algorithm 1: SmartRef frame-processing loop

```
Input: video stream S, court boundary polygon B, net region N
for each frame f in S do
    f' ← preprocess(f)
    D ← detect(f')            // shuttlecock, players
    T ← track(D)              // persistent IDs
    for each tracked shuttlecock b in T do
        c ← centre(b)
        if c is outside B then raise OUT event
    end for
    if net rule on N is violated then raise NET event
    if an event was raised then
        activate buzzer and LED, draw overlay
        log(timestamp, event), update dashboard
end for
```

E. Alerts and Visual Feedback

When a violation is detected, the buzzer and LED are driven through GPIO, the decision (for example, “Ball Out”) is overlaid on the video frame together with the drawn court boundary, and a timestamped entry is written to the event log. The system runs entirely on the Raspberry Pi and is optimised for low latency, so that the delay between an event and its indication stays short.

F. Web Dashboard and Replay

The Flask-based dashboard is opened from a browser on the local network, through SSH tunnelling or through a remote-desktop session. Its development followed the steps shown in Fig. 5, which are connecting to the camera, processing the video input, detecting events, displaying the live feed, showing real-time scores, adding a replay screen and finally running the application on the Raspberry Pi.

The finished dashboard has a live video panel (camera stream or uploaded video), a match timer with start, pause and stop controls, a scoreboard for the 2 sides, a timestamped event log, a replay screen that lets the user review the footage around a detected event, and a control panel for switching between live and uploaded input, starting or stopping the analysis and adjusting the court calibration. Replay allows a human referee to confirm or overrule any automatic decision. Fig. 6 shows the control panel of the implemented dashboard.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, November 2026
Impact Factor: 8.2

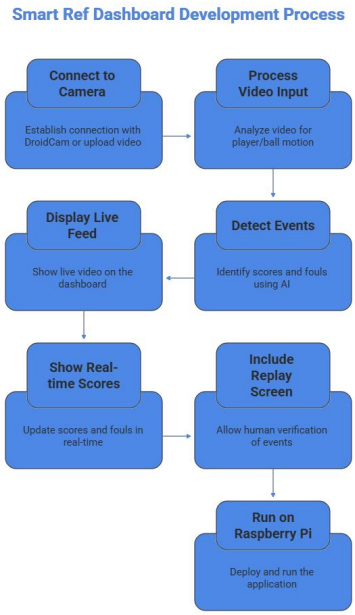

Fig. 5. Development process of the SmartRef dashboard.

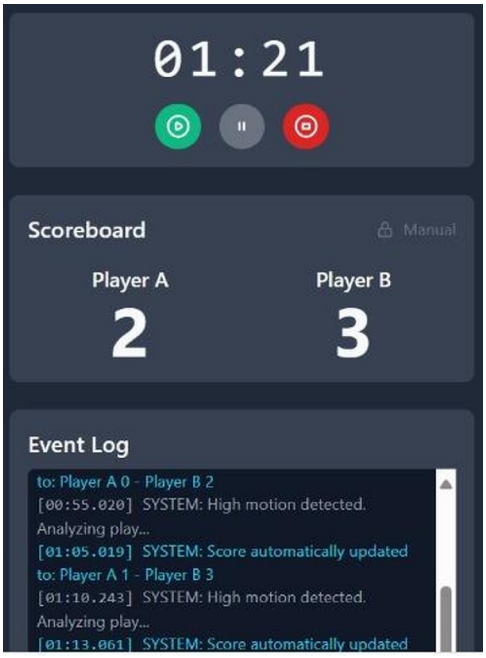

Fig. 6. Dashboard control panel with timer, scoreboard and event log.

G. Implementation on the Raspberry Pi

The back end runs on the Pi and handles video capture over RTSP, object detection with YOLO, tracking, rule checking and the GPIO alerts. Listing 1 gives the core Python routine. It loads the YOLOv8n model, reads frames from the IP camera, takes the centre of every detected ball, tests it against the court boundary with the point-in-polygon function of OpenCV, and switches on the buzzer and LED and writes “Ball Out!” on the frame when the ball is outside the court. The court boundary is drawn on every frame so that the referee can see the calibrated region, and all outputs are reset when the program stops.

Listing 1: Core SmartRef routine in Python

```
import cv2
import numpy as np
from ultralytics import YOLO
import RPi.GPIO as GPIO

# Initialize GPIO for buzzer/LED
GPIO.setmode(GPIO.BCM)
BUZZER_PIN = 18
LED_PIN = 23
GPIO.setup(BUZZER_PIN, GPIO.OUT)
GPIO.setup(LED_PIN, GPIO.OUT)

# Load YOLO model
model = YOLO('yolov8n.pt')
```

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, November 2026
Impact Factor: 8.2

```
# Capture video from IP camera
cap = cv2.VideoCapture('rtsp://<camera_ip>:554/stream')

# Court boundary coordinates (example)
BOUNDARY = np.array([(100, 100), (500, 100), (500, 400),
(100, 400)], dtype=np.int32)

while True:
    ret, frame = cap.read()
    if not ret:
        break
    # Detect objects (ball, players)
    results = model(frame)
    detections = results[0].boxes.data.cpu().numpy()  # [x1,
y1, x2, y2, conf, class]
    # Check ball position against boundary
    for det in detections:
        if det[5] == 0:  # Assuming class 0 is ball
            x, y = (det[0] + det[2]) / 2, (det[1] + det[3])
/ 2  #Center of ball
            if not cv2.pointPolygonTest(BOUNDARY, (float(x),
float(y)), False) > 0:
GPIO.output(BUZZER_PIN, GPIO.HIGH)  # Trigger buzzer
GPIO.output(LED_PIN, GPIO.HIGH)     # Trigger LED
                cv2.putText(frame, "Ball Out!", (50, 50),
                            cv2.FONT_HERSHEY_SIMPLEX, 1, (0,
0, 255), 2)
print("Violation: Ball Out")
    # Display frame with overlays
    cv2.polylines(frame, [BOUNDARY], True, (0, 255, 0), 2)
    cv2.imshow('SmartRef', frame)
    # Exit on key press
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Cleanup
GPIO.output(BUZZER_PIN, GPIO.LOW)
GPIO.output(LED_PIN, GPIO.LOW)
cap.release()
cv2.destroyAllWindows()
GPIO.cleanup()
```

H. Dataset and Logged Data

The dataset for SmartRef consists of badminton video clips, which were used to train the YOLO models. During the experimental tests, the system also logs the original video frames, the detected objects (ball and players), the rule violation events such as ball out and net touch, and the timestamped alerts together with the accuracy metrics. These logs were used to work out the results reported in Section VI.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, November 2026
Impact Factor: 8.2

V. EXPERIMENTAL SET-UP

The system was validated through software simulation and hardware prototyping. OpenCV and TensorFlow Lite were used to test video processing and model performance on the Raspberry Pi 4B, and badminton video clips served as the dataset for training and evaluating the YOLOv5n/YOLOv8n detectors. The complete prototype, made up of the Raspberry Pi 4B, the network camera and the GPIO-connected buzzer and LED, was then tested in controlled indoor and outdoor environments. The camera was calibrated for the court boundaries before each experiment, and the tests were repeated under different lighting conditions to check reliability.

The experiments focused on detecting ball in/out events and net touches in badminton. The following measures were recorded with Python scripts and serial monitor output, namely (i) detection accuracy for ball in/out events over 50 test cases, (ii) processing time per frame, (iii) the reliable operating range from the camera, (iv) false positives and their causes, (v) the reliability of the buzzer and LED alert, (vi) power use during operation, and (vii) the speed and consistency of the calls compared with traditional human officiating. Accuracy is computed as

$$\text{Accuracy} = \frac{N_{\text{correct}}}{N_{\text{total}}} \times 100\% \qquad (2)$$

where N_{correct} is the number of correctly classified test cases and N_{total} is the total number of test cases. The frame rate that corresponds to a processing time t in milliseconds is

$$\text{FPS} \approx \frac{1000}{t} \qquad (3)$$

VI. RESULTS AND DISCUSSION

A. Quantitative Results

Table III summarises the measured performance. SmartRef correctly classified 46 of the 50 ball in/out test cases, which is an accuracy of 92% by (2). The average processing time was 50 to 80 ms per frame, which by (3) works out to roughly 12.5 to 20 frames per second and was enough for real-time indication of violations in the tested scenarios. Video streaming and detection stayed stable up to 10 m from the camera in both indoor and outdoor settings, and the buzzer and LED responded to every detected violation in the controlled tests. Only 2 to 3 false positives were seen, and these were caused by occlusions or fast-moving objects.

The system also outperformed traditional human officiating in speed and consistency, particularly when detecting ball in/out events. Optimised processing and sleep cycles kept the power use low, which makes the set-up suitable for portable use, and the whole system is built from low-cost, off-the-shelf parts. Fig. 7 shows the dashboard while a match video is being analysed, with the live video, the match timer, the scoreboard and the event log updating together.

TABLE III: Measured performance of SmartRef

Metric	Result
Ball in/out detection accuracy	92% (46 of 50 test cases)
Processing time per frame	50 to 80 ms (about 12.5 to 20 FPS)
Stable operating range	Up to 10 m from the camera, indoors and outdoors
False positives	2 to 3, due to occlusions or fast-moving objects
Buzzer/LED alert reliability	100% in controlled tests
Low-light operation	Minor errors observed
Power consumption	Low, with optimised processing and sleep cycles
Comparison with human officiating	Faster and more consistent, especially for ball in/out calls
Hardware cost	Low, built from off-the-shelf components

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, November 2026
Impact Factor: 8.2

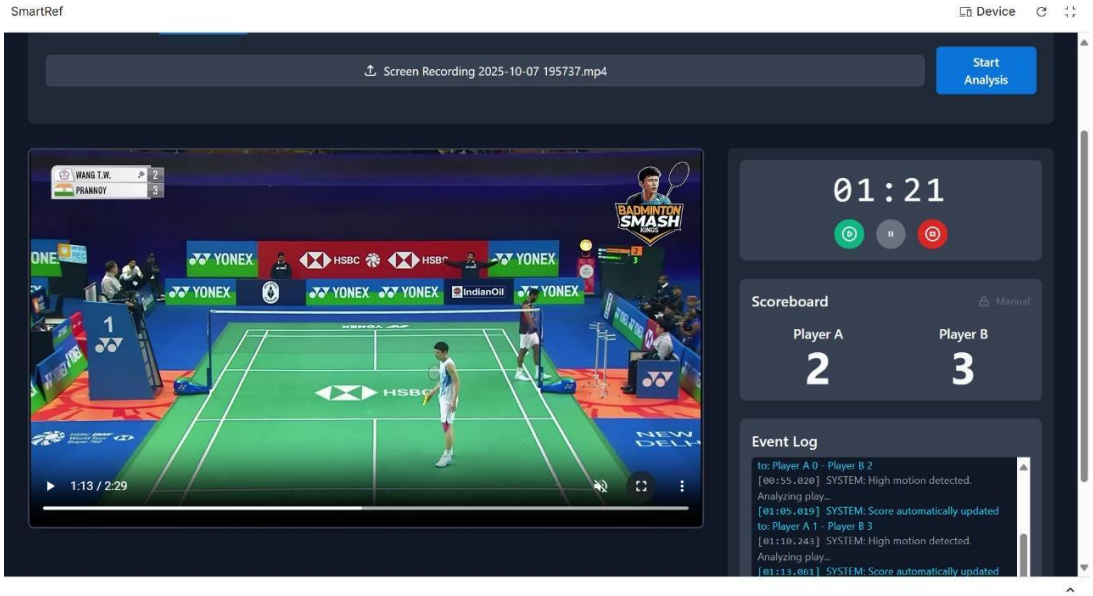

Fig. 7. SmartRef dashboard during the analysis of a badminton match video.

B. Discussion

The results show that a complete vision-based officiating pipeline can run on a single Raspberry Pi 4B with accuracy and latency that are good enough for amateur use. The 4 incorrect cases were mainly linked to 2 factors, which were occlusion of the shuttlecock by the players or the racket, and fast motion that blurs or shifts the small shuttlecock between frames. Low light also caused minor errors, because the detection confidence for a small, low-contrast object falls quickly as the light drops. The offline design, the lack of any cloud dependency and the use of off-the-shelf parts are the main practical advantages over the systems in Table I. Compared with human officiating, the system made its calls faster and more consistently, and the replay function of the dashboard still lets a human referee check any automatic call, which covers the remaining error rate.

C. Applications

SmartRef can be used in several settings. In grassroots tournaments it gives affordable officiating for school and college sports such as badminton, and in training sessions it helps coaches to monitor rule adherence and improve player performance. Amateur leagues with small budgets can use it to make local matches fairer, and the logged game events can be used for post-match analysis and strategy. Event organisers can use it to keep officiating consistent and unbiased across a tournament, and the project itself is a useful learning platform for students working with AI, computer vision and embedded systems.

D. Limitations

The present evaluation has some limits that should be kept in mind. The quantitative study is based on 50 ball in/out cases from a single sport, and numbers for net events and automatic scoring are not reported here. The rule checker works in image coordinates from a single camera, so it does not correct for perspective or shuttle height, and it has no tolerance margin for shuttles that land on a line. The comparison with human officiating was made in controlled tests, and a larger trial in real tournaments would make it stronger. SmartRef should therefore be seen as an officiating aid for grassroots play rather than as a replacement for certified systems.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, November 2026
Impact Factor: 8.2

VII. CONCLUSION AND FUTURE WORK

This paper presented SmartRef, a low-cost, portable and fully offline vision-based referee system that runs on a Raspberry Pi 4 Model B. By combining a nano YOLO detector, SORT/DeepSORT tracking, a calibrated geometric rule checker, GPIO alerts and a replay-capable web dashboard, the system detected ball in/out events with 92% accuracy on 50 test cases at 50 to 80 ms per frame, worked reliably up to 10 m from the camera and raised every alert in the controlled tests. It was faster and more consistent than human officiating for ball in/out calls and kept its power use low. Occlusion, fast motion and low light remain the main sources of error. The modular design makes the approach suitable for school, college and local-club tournaments.

Future work will address the limitations above and extend the system in the following ways.

1) Multi-camera integration with homography-based court mapping and 3-D analysis for better coverage and landing-point accuracy.
2) Wearable or on-court sensors to detect player contact and foul intensity.
3) Larger detection models such as YOLOv8m with hardware accelerators such as the Coral Edge TPU.
4) IoT integration for remote monitoring and analytics.
5) Extension to other sports such as basketball and cricket.
6) A larger annotated dataset and a tournament-scale comparison with human referees.

ACKNOWLEDGMENT

The authors thank Shri P. Sriram, Chairman, and Dr. A. Ramesh, Principal, Chennai Institute of Technology, for their support and encouragement. They are grateful to their project guide and Head of the Department, Dr. K. Lakshmi Prabha, Department of Electronics and Communication (Advanced Communication Technology), Chennai Institute of Technology, for her guidance throughout this work, and to the Program Coordinator, Mrs. A. Tamilselvi, and the Project Coordinator, Mrs. R. Rajasree, for their help.

REFERENCES

[1] N. Owens, C. Harris, and C. Stennett, "Hawk-Eye tennis system," in *Proc. Int. Conf. Visual Information Engineering (VIE)*, Guildford, U.K., 2003, pp. 182-185.

[2] M. Kopania, J. Nowisz, and A. Przelaskowski, "Automatic shuttlecock fall detection system in or out of a court in badminton games: Challenges, problems, and solutions from a practical point of view," *Sensors*, vol. 22, no. 21, Art. no. 8098, 2022, doi: 10.3390/s22218098.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779-788.

[4] G. Jocher, A. Chaurasia, and J. Qiu, *Ultralytics YOLOv8*, 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[5] G. Jocher, *Ultralytics YOLOv5*, 2020. [Online]. Available: https://github.com/ultralytics/yolov5

[6] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, 2016, pp. 3464-3468.

[7] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, 2017, pp. 3645-3649.

[8] G. Thomas, R. Gade, T. B. Moeslund, P. Carr, and A. Hilton, "Computer vision for sports: Current applications and research topics," *Comput. Vis. Image Underst.*, vol. 159, pp. 3-18, 2017.

[9] B. T. Naik, M. F. Hashmi, and N. D. Bokde, "A comprehensive review of computer vision in sports: Open issues, future trends and research directions," *Appl. Sci.*, vol. 12, no. 9, Art. no. 4429, 2022, doi: 10.3390/app12094429.

[10] Y.-C. Huang, I.-N. Liao, C.-H. Chen, T.-U. İk, and W.-C. Peng, "TrackNet: A deep learning network for tracking high-speed and tiny objects in sports applications," in *Proc. IEEE Int. Conf. Advanced Video and Signal Based Surveillance (AVSS)*, 2019, pp. 1-8.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, November 2026
Impact Factor: 8.2

[11] R. Voeikov, N. Falaleev, and R. Baikulov, "TTNet: Real-time temporal and spatial video analysis of table tennis," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020, pp. 3866-3874.
[12] G. L. Goh, G. D. Goh, J. W. Pan, P. S. P. Teng, and P. W. Kong, "Automated service height fault detection using computer vision and machine learning for badminton matches," *Sensors*, vol. 23, no. 24, Art. no. 9759, 2023, doi: 10.3390/s23249759.
[13] J. Zhao, C. Zhu, B. Leng, and J. Qi, "Design and development of a public AI referee assistance system based on Harmony OS platform," *Sensors*, vol. 25, no. 7, Art. no. 2127, 2025, doi: 10.3390/s25072127.
[14] Raspberry Pi Ltd., *Raspberry Pi 4 Model B Datasheet*, 2019. [Online]. Available: https://datasheets.raspberrypi.com
[15] G. Bradski, "The OpenCV library," *Dr. Dobb's Journal of Software Tools*, 2000.
[16] Pallets Projects, *Flask Documentation*. [Online]. Available: https://flask.palletsprojects.com
[17] Google, *TensorFlow Lite (LiteRT) Documentation*. [Online]. Available: https://www.tensorflow.org/lite