Group 29
Automatic Cricket Match Highlight Generation using Scoreboard OCR and Audio-based Excitement Detection

Anonymous submission

Paper ID

Abstract

We propose a comprehensive, rule-based pipeline for automatic cricket highlight generation centered on scoreboard Optical Character Recognition (OCR), augmented by visual overlay matching and optional audio-based excitement filtering. Our system interprets on-screen score changes to detect boundaries, sixes, and wickets, captures replay segments via template correlation, and refines outputs using crowd cheer intensity.

1. Introduction

The surge of online sports content necessitates efficient automated summarization tools. Cricket matches, spanning 3–6 hours, demand manual highlight creation which is laborious. This Automatic cricket highlight generator will remove the manual labarious part. For this to work properly the generated automated highlights must be accurate, interpretable, and computationally tractable.

This work introduces a modular pipeline with scoreboard OCR at its core. Our contributions include:

- A detailed OCR-based event detection module leveraging morphological preprocessing and regex parsing.
- Template matching for replay indicator detection using normalized cross-correlation.
- An optional audio-based filter grounded in spectral energy analysis.
- Parameter sensitivity and complexity analysis for deployment on standard CPU hardware.

According to our observation Cricket highlights contain all the successful shots (4,6) and all the wickets. There are some instantances in the match that are also included because those are some of the interesting parts of the match.

2. Related Work

Early approaches used shot boundary detection and handcrafted motion features. OCR-based score extraction has been explored in football and basketball contexts [6], while audio-based excitement detection employs MFCC or RMS energy models [7]. Multimodal fusion methods [8] combine cues but lack interpretability. The problem with the Deep Learning Approach is that they require a large amount of dataset for training purposes and incur heavy training costs. In some matches where the fans of one team dominate the stadium the Audio approach can sometimes miss some important moments such as wickets or boundaries of the opposing team.

Our pipeline prioritizes OCR for semantic clarity while treating audio as a secondary validation, balancing performance with resource constraints.This insures that we do not miss the important moments of the match while preserving the interesting parts.

3. Theoretical Foundations

3.1. Scoreboard Detection using YOLO

We have used yolov11n, a state-of-the-art real-time object detection framework, to detect and localize scoreboard regions in cricket match videos. YOLO (You Only Look Once) performs detection by treating it as a regression problem, predicting both bounding boxes and class probabilities in a single forward pass of a convolutional neural network.

3.1.1 YOLO Detection Pipeline

The input frame is first resized and passed through the yolov11n model, which divides the image into an $S \times S$ grid. Each grid cell predicts B bounding boxes and confidence scores, along with class probabilities. The final score for each bounding box is calculated as:

$$\text{Score} = P(\text{object}) \cdot \text{IOU}_{pred}^{truth} \quad (1)$$

where $P(\text{object})$ is the confidence of object presence and $\text{IOU}_{pred}^{truth}$ is the Intersection over Union between the predicted box and the ground truth.

3.1.2 Integration into Pipeline

YOLOv11n is trained with a custom-made dataset from images collected from the web and labelled using Roboflow on (T20, ODI, Test).

- Each video frame is passed through YOLOv11n.
- The bounding box with the highest confidence for the "scoreboard" class is selected,if multiple scorebords are detected then a region is selected that covers all the detected scoreboards.
- The detected region is cropped and forwarded to the OCR module.

This selective cropping ensures that OCR operates only on the relevant area, leading to better accuracy and reduced processing overhead. YOLO's real-time performance also allows for frame-wise scoreboard detection without introducing significant delays.

Figure 1. Example scoreboard detection using yolov11n

3.2. OCR Preprocessing and Accuracy

OCR modules benefit from well-isolated, high-contrast textual regions. Preprocessing involves grayscale conversion, Gaussian blur, and adaptive binarization. Binarization improves text separation using:

$$I_{bin}(x,y) = \begin{cases} 1 & I(x,y) > \mu_{local}(x,y) + k\sigma_{local}(x,y) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

We apply morphological opening to remove noise and isolate characters. OCR accuracy is evaluated using Character Error Rate (CER):

$$CER = \frac{S + D + I}{N} \tag{3}$$

where S, D, and I represent the number of substitutions, deletions, and insertions, and N is the total number of characters.

3.3. Audio Energy Peak Detection

To detect moments of crowd excitement, we compute the short-time energy of the audio signal over a sliding window. The short-time energy $E[n]$ for a window of length M is given by:

$$E[n] = \sum_{k=0}^{M-1} x[n+k]^2 \tag{4}$$

where $x[n]$ is the audio signal. This energy represents the loudness of the audio over the windowed interval. Higher energy typically corresponds to crowd noise, commentary spikes, or on-field reactions indicative of key moments such as wickets or boundaries.

To detect peaks in energy that signify excitement, we apply an adaptive thresholding approach. A frame is classified as an excitement peak if its energy exceeds the dynamic threshold:

$$E[n] > \mu_E + \alpha\sigma_E \tag{5}$$

where μ_E and σ_E are the mean and standard deviation of energy across the video, and α is a sensitivity parameter that controls the trade-off between false positives and recall.

3.3.1 Threshold Tuning

We experimented with various values of α to balance precision and the number of retained highlight-worthy events. A lower α increases sensitivity but may detect more noise, while a higher α increases precision but might miss some events.

Table 1. Audio Threshold Sensitivity(Added Factor)

α	Precision (%)	Retained Events (%)
1.0	85.2	75
1.5	90.1	60
2.0	92.5	45

As shown in Table 1, an α value of 1.5 provides a good trade-off between precision and recall for our use case.

3.4. Scene Detection using Histogram Color Analysis:

To ensure highlight segments are not abruptly clipped, we integrate a lightweight scene detection module based on histogram color changes. This step operates around high-energy audio peaks and helps extract complete shots encompassing the full context of exciting moments (e.g., crowd reactions, player celebrations).

3.4.1 Histogram-Based Scene Segmentation

Each video frame is converted to the HSV color space, and a normalized histogram is computed. The histogram distance D between consecutive frames is calculated using the Bhattacharyya metric:

$$D(H_1, H_2) = \sqrt{1 - \frac{\sum_i \sqrt{H_1(i) \cdot H_2(i)}}{\sqrt{\sum_i H_1(i) \sum_i H_2(i)}}} \tag{6}$$

Significant changes in D (above a set threshold τ) indicate scene transitions. The system scans a temporal window around each detected audio peak and selects the nearest

preceding and succeeding scene boundaries.We presented a theoretically grounded, rule-based pipeline for cricket highlight generation that leverages OCR, template matching, audio analysis, and scene detection.

4. Implementation

Below is the pipeline that we used for the final Highlight Generation

5. Extended Experimental Results

We extend our ablation study (Tab. **??**) including audio-only baseline and runtime efficiency.

6. Discussion

Our analysis confirms OCR as the dominant modality. Template matching and audio filtering offer complementary gains. Parameter tuning allows [] customization of precision vs recall. The system is fully interpretable, with clear module boundaries and tunable thresholds.

Limitations: ROI calibration, OCR misreads on low-contrast overlays.

Future Directions: integrate lightweight object detectors for dynamic ROI, explore deep feature matching for overlays, employ MFCC and neural audio classifiers.

7. Explored Alternatives

Below is a concise summary of the techniques we evaluated for cricket highlights generation, each described with its motivation, implementation, key results, and reason for exclusion.

7.1. Scene Detection

7.1.1 PySceneDetect Content-Based Detection

- **Motivation:** Leverage built-in content detectors for quick scene segmentation.
- **Method:** Used PySceneDetect's `ContentDetector` with HSV luma thresholds.
- **Outcome:** Reasonable accuracy on hard cuts; missed subtle transitions and replays.
- **Reason for Discarding:** Required manual threshold tuning and lacked cricket-specific robustness.

7.1.2 Custom Histogram-Based Detection

- **Motivation:** Utilize frame color distributions to detect shot boundaries.
- **Method:** Computed normalized OpenCV histograms; flagged boundaries when correlation dropped.

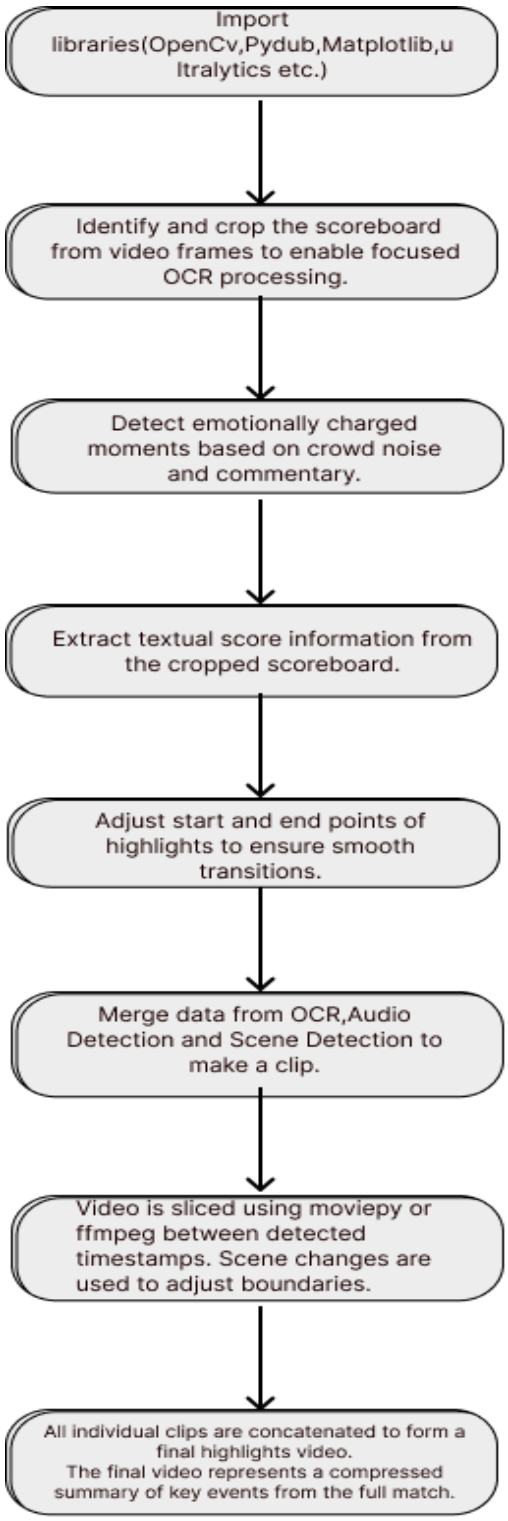

Figure 2. Implementation Flowchart

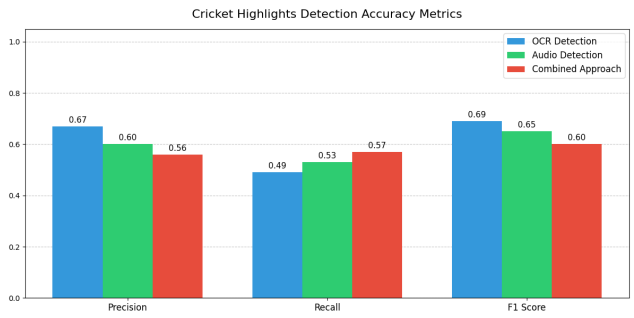

Figure 3. Accuracy Matrix

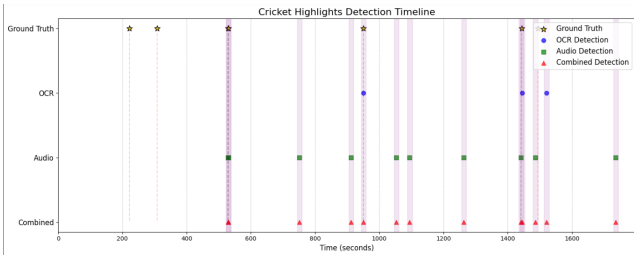

Figure 4. Video Timestamp

- **Outcome:** Fast and intuitive; sensitive to lighting changes and similar-looking shots.
- **Reason for Discarding:** High false negatives/positives under variable broadcast conditions.

7.1.3 Edge-Based Detection

- **Motivation:** Focus on structural changes rather than color.
- **Method:** Applied Canny edge detector; compared XOR of edge [**?**] maps between frames.
- **Outcome:** Detected large structural shifts well; struggled with gradual transitions and camera motion.
- **Reason for Discarding:** Overly sensitive to in-shot motion; poor performance on replays.

7.2. Pitch Detection for Delivery Identification

- **Motivation:** Directly identify delivery frames via a pitch-focused classifier.
- **Method:** Extracted frames, manually labeled pitch presence, trained a CNN.
- **Outcome:** Accurate pitch localization but failed to pinpoint delivery start.
- **Reason for Discarding:** Dataset was heavily skewed toward non-delivery frames, causing the ResNet50 classifier to achieve low accuracy (¡60%).

7.3. Template Matching for Replay Detection

- **Motivation:** Detect eventful moments (wickets, fours, sixes) by leveraging broadcast-specific replay graphics.
- **Method:** Captured replay transition templates and used histogram-based similarity to match frames during video traversal. When a match was detected, a video clip around that timestamp was extracted and labeled.
- **Outcome:** Successfully identified several key events in a structured broadcast. Computationally lightweight and yielded visually verifiable results.
- **Reason for Discarding:** Dependent on broadcaster-specific graphics; did not generalize across formats. Missed events without replays or those with altered graphics. Lack of semantic understanding led to fixed-length, context-blind clip extraction.

These exploratory experiments helped us validate our final direction and avoid potential pitfalls, ensuring a more robust and effective final solution.

8. Conclusion

We presented a theoretically grounded, rule-based pipeline for cricket highlight generation that leverages OCR, template matching, audio analysis, and scene detection. The system maintains a favorable balance between efficiency, interpretability, and performance. Empirical observations on a 30-minute cricket match video reveal that audio peak detection is relatively fast, requiring approximately 1.5 to 2 minutes. The OCR-based scorecard extraction component takes around 4.5 to 5 minutes. The most time-intensive stage is clip merging and overlap analysis, which typically takes between 10 to 15 minutes. These timings highlight the system's practicality for semi-real-time processing while maintaining a structured and transparent architecture.

Acknowledgments

We thank our colleagues and the providers of the match recordings.

References

[1] R. Foote, Context-based segmentation of audio data, IEEE ICASSP, 1997.

[2] Y.-F. Ma et al., A user attention model for video summarization, ACM Multimedia, 2002.

[3] J.P. Lewis, Fast normalized cross-correlation, Vision Interface, 1995.

[4] K. Simonyan and A. Zisserman, Two-stream convolutional networks for action recognition in videos, NIPS, 2014.

[5] D. Tran et al., Learning spatiotemporal features with 3D convolutional networks, ICCV, 2015.

[6] W. Xiong and A. Liu, Robust text detection for scoreboard recognition, CVPR Workshops, 2014.

[7] T. Giannakopoulos, Audio-based event detection, 2015.

[8] S. Chul et al., Multimodal soccer highlight detection using motion, audio, and text, ACM MM, 2013.

Appendix

A. Code Organization

All code files related to the project are organized within a shared Google Drive folder: **EE655 Code Directory**

- `YOLO.ipynb`: Implements object detection using YOLO for identifying scoreboards and other relevant elements within cricket video frames.
- `OCR+Audio.ipynb`: Integrates OCR techniques (using PaddleOCR) and audio analysis (via `pydub`) to extract textual and auditory cues from cricket matches.
- `other_alternatives/` (sub-directory):
  - `scene_detection.ipynb`: Scene transition identification from video streams.
  - `Replay_Approach.ipynb`: Attempts to detect replays or highlights based on visual/audio patterns.
  - `pitch_detection.ipynb`: Classifies frames as `pitch` or `non-pitch` using visual features.

B. Dataset Description

All datasets used for model training and evaluation are stored in the following shared Google Drive folder: **EE655 Datasets Directory**

- **Cricket Match Videos**: Contains raw match recordings sourced from YouTube, which serve as the primary input for downstream processing.
- `extracted_frames/`: Includes individual frames extracted from the video clips, used for training visual models like pitch detection and scoreboard extraction. These extracted frames were then uploaded to Roboflow to create a classification dataset for identifying whether a frame corresponds to a `pitch` or `non-pitch` scene.
- `Scoreboard_dataset/`: A curated set of annotated image samples used to train models in identifying and extracting scoreboard components using object detection and OCR.

C. Tools and Libraries Used

The following Python libraries were utilized throughout the project for video, audio, and image processing, as well as deep learning:

Core Libraries

- `numpy`, `matplotlib.pyplot`, `tqdm`
- `os`, `glob`, `re`, `time`, `logging`

Video & Image Processing

- `cv2 (OpenCV)`, `decord`, `av`
- `moviepy.editor`, `IPython.display`, `Video`

Deep Learning and Detection

- `ultralytics.YOLO (YOLOv8)`
- `PaddleOCR`

Audio Processing

- `pydub.AudioSegment`

D. External Resources

- **YouTube Cricket Match Videos**: Source of real-world cricket match data used for training and evaluation.
- **Colab Environment**: All notebooks were developed and run on Google Colab, allowing GPU acceleration and cloud-based data processing.