Cricket Ball Detection and Tracking from a Single Static Camera

EdgeFleet.AI – Computer Vision Assessment

1. Introduction

Tracking a cricket ball in match footage is a challenging computer vision problem due to the ball's small size, high speed, motion blur, and frequent occlusions. This project presents a complete end-to-end system to detect and track a cricket ball from videos recorded using a single, fixed camera, as required by EdgeFleet.AI.

The system detects the cricket ball centroid in each frame where it is visible, maintains a continuous trajectory across frames, and produces both per-frame annotation files and processed videos with trajectory overlays. The focus of this project is on robustness, reproducibility, and clear handling of detection failures.

2. Problem Statement

The objective of this project is to build a computer vision pipeline that:

- Detects the cricket ball centroid in each video frame
- Handles frames where the ball is not visible
- Tracks the ball trajectory continuously over time

Outputs:

- A CSV annotation file with frame index, centroid coordinates, and visibility flag
- A processed video with centroid and trajectory overlay
- Works on videos captured from a single static camera
- Is fully reproducible using provided code and model files

3. System Overview

The proposed system follows a modular, sequential pipeline:

1.Video Input:-

A cricket video recorded from a single fixed camera is provided as input.

2.Motion-Based Detection:-

Frame differencing is used to identify fast-moving objects and generate candidate ball detections.

3.YOLO-Based Fallback Detection:-

A pretrained YOLOv8 model is used when motion detection fails.

4.Kalman Filter Tracking:-

A Kalman filter maintains a smooth and continuous trajectory across frames, even during short detection failures.

5.Annotation Generation:-

Per-frame centroid and visibility information is written to a CSV file.

6.Visualization:-

Bounding box, centroid, and trajectory are drawn on each frame to generate a processed output video.

4. Modelling Decisions

4.1 Motion-Based Detection (Primary Detector):-

A motion-based detector using frame differencing was chosen as the primary detection method because:

- The cricket ball is a fast-moving object
- Motion cues remain effective even under motion blur
- It is computationally lightweight and fast
- It works well when background remains static
- Area and radius filtering are applied to reduce noise and false positives.

4.2 YOLOv8 Detection (Fallback):-

A pretrained YOLOv8 object detector is used as a fallback when motion detection fails. The reasons for this choice are:

- Robust detection under complex backgrounds
- Ability to recover detections missed by motion-based methods
- No training required on the provided dataset
- Only detections within a valid size range are considered to filter out irrelevant objects.

4.3 Kalman Filter Tracking:-

A Kalman filter is used to track the cricket ball centroid across frames. This choice enables:

- Smooth trajectory estimation
- Continuous prediction when detections are temporarily missing
- Noise reduction in centroid coordinates
- The tracker predicts the ball’s position even when no valid detection is available, ensuring trajectory continuity.

4.4 Centroid-Based Representation:-

The system tracks the centroid of the cricket ball instead of full bounding boxes because:

- The required output format is centroid-based
- Centroid tracking is simpler and more stable for small objects
- It reduces noise from bounding box size variations

5. Fallback Logic and Visibility Handling

The system implements explicit fallback logic to ensure robustness:

Motion Detection Attempt:-

If a valid motion-based detection is found, it is used.

YOLO Fallback:-

If motion detection fails, YOLOv8 detection is triggered.

Tracker Prediction:-

If both detectors fail, the Kalman filter predicts the next position.

Visibility Flag Assignment:-

- visible = 1 → detection available
- visible = 0 → detection missing, prediction used
- When the ball is not visible, centroid values are recorded as -1, -1 in the annotation file.

6. Annotation Output Format

The system generates a CSV file containing per-frame annotations in the following format:

```
frame, x, y, visible
0,512.3,298.1,1
1,518.7,305.4,1
2,-1,-1,0
```

This format ensures clarity and compatibility with downstream analysis or evaluation pipelines.

7. Dataset Usage Declaration

The dataset provided by EdgeFleet.AI is used strictly for testing and evaluation purposes only.

No training or fine-tuning is performed using this dataset.

8. Assumptions

- The system is designed under the following assumptions:
- Videos are recorded from a single fixed camera
- The background remains mostly static
- The cricket ball occupies a limited size range
- Temporary occlusions and missed detections are acceptable
- No camera motion or zoom is present

9. Issues Faced and Improvements

Issues Faced

- Missed detections during fast ball motion
- Noise in early frames due to background motion
- Occasional false positives from moving players or shadows

Improvements Implemented

- Area and radius filtering for motion detections
- Confidence-based filtering for YOLO detections
- Distance-based validation (MAX_JUMP) in Kalman updates
- Continuous prediction to maintain trajectory during occlusions
- These improvements significantly enhanced trajectory stability and reduced detection noise.

10. Reproducibility

The project is fully reproducible due to:

- Centralized configuration parameters
- Included pretrained model file
- Deterministic inference pipeline
- Clear command-line execution instructions
- Modular and readable code structure

11. Conclusion

This project demonstrates a robust and reproducible approach to cricket ball detection and tracking using a combination of motion-based detection, deep learning-based fallback, and Kalman filter tracking. The system successfully meets all EdgeFleet.AI requirements by producing accurate per-frame annotations and clear trajectory-overlayed videos while handling detection failures gracefully.

12. Example Outputs

- Processed MP4 videos with centroid and trajectory overlay are stored in the results/ directory.

- Per-frame CSV annotation files are stored in the annotations/ directory.