Cricket Ball Detection and Tracking from a Single Static Camera

Kunal Kamalkishor Bhosikar

1 Introduction

Detecting and tracking a cricket ball in broadcast-style videos is a challenging computer vision problem due to the ball's small size, high speed, motion blur, frequent occlusions, and visual similarity to other moving objects such as bats and player highlights.

This report describes a complete, reproducible computer vision pipeline for detecting and tracking a cricket ball from videos captured using a single static camera, as required by the EdgeFleet.ai AI/ML assessment. The system outputs per-frame ball centroids with visibility flags, along with a processed video visualizing the estimated ball trajectory.

2 Problem Setup and Constraints

The task requires:

- Detection of the cricket ball centroid in each frame where it is visible
- Per-frame annotations in CSV format: `frame, x, y, visible`
- A processed video with centroid and trajectory overlay
- Fully reproducible code for inference and evaluation

Key constraints:

- Single, fixed camera
- Provided dataset must **not** be used for training

Given these constraints, a classical computer vision pipeline with temporal filtering was chosen over data-driven deep learning approaches.

3 System Overview

The final system consists of the following stages:

1. Frame-by-frame video processing
2. Motion-based candidate extraction
3. Color, size, and shape filtering
4. Kalman filter–based tracking
5. Visibility reasoning and output generation

4 Ball Detection

4.1 Motion Detection

Since the camera is static, motion is a strong cue for identifying candidate ball regions. Motion is extracted using frame differencing / background subtraction followed by morphological operations to remove noise.

4.2 Color Filtering

Initial experiments showed that motion detection alone highlighted players, bats, and other moving objects. To reduce false positives, HSV color filtering was applied.

For white-ball scenarios, the following HSV range was used:

$$H \in [0, 180],\ S \in [0, 60],\ V \in [170, 255]$$

The final foreground mask is obtained by:

$$\text{Foreground} = \text{Motion Mask} \cap \text{Color Mask}$$

4.3 Shape and Size Constraints

Candidate contours are filtered using:

- Area constraints (small objects only)
- Circularity measure:

$$C = \frac{4\pi A}{P^2}$$

Only compact, near-circular contours are considered valid ball candidates.

4.4 Failure Modes Observed

Initial detection suffered from:

- Player outlines being detected as moving blobs
- Bat edges falsely classified as circular objects
- Motion blur causing ball shape distortion

These were addressed by progressively tightening and then carefully relaxing detection thresholds.

5 Tracking Using Kalman Filter

5.1 State Representation

A Kalman filter with a constant-velocity model is used. The state vector is:

$$\mathbf{x} = [x, y, v_x, v_y]^T$$

5.2 Delayed Initialization

A key issue encountered was incorrect tracker initialization due to early false detections. Initializing the Kalman filter on the first detection caused the tracker to lock onto incorrect positions, leading to drift.

To address this, the tracker is initialized only after **multiple consistent detections** (typically 3 consecutive detections). During this phase, no prediction step is executed.

5.3 Prediction and Correction

Once initialized:

- The Kalman filter predicts the ball position every frame
- If a detection is available, the filter is corrected
- If detection is missing, prediction is used for short gaps

5.4 Detection Gating

To prevent false detections from corrupting the tracker, a distance-based gating mechanism is applied:

$$(x_d - x_p)^2 + (y_d - y_p)^2 < T$$

where (x_d, y_d) is the detected centroid and (x_p, y_p) is the predicted centroid.

Detections violating this constraint are rejected.

6 Visibility Logic

Visibility is defined as:

- `visible = 1` if the ball is detected or confidently predicted
- `visible = 0` if the ball is lost for an extended duration

Short occlusions and motion blur are handled by allowing prediction-based visibility for a fixed number of frames.

7 Trajectory Visualization

The final output video overlays:

- A green dot at the current estimated ball centroid
- A blue polyline showing the recent ball trajectory

The trajectory is reset whenever the ball is lost to avoid visually incorrect long-range connections.

8 Results and Observations

- Motion-only detection produced excessive false positives
- Adding color and shape constraints significantly reduced noise
- Overly strict detection reduced recall and broke trajectory continuity
- Kalman prediction restored smooth trajectories despite sparse detections

The final system produces stable and visually coherent ball trajectories across most frames where the ball is visible.

9 Limitations

- Performance degrades under extreme motion blur
- White bats in close proximity to the ball remain challenging
- No explicit bounce modeling is included
- HSV thresholds may require tuning for different lighting conditions

10 Conclusion

This work demonstrates that a carefully engineered classical computer vision pipeline, combined with temporal filtering, can robustly detect and track a cricket ball in static-camera videos without any training data.

The systematic debugging process—analyzing failure modes, visualizing intermediate representations, and incrementally refining constraints—proved critical to achieving stable performance. The final system satisfies all task requirements while remaining fully reproducible and computationally efficient.