Cricket Ball Detection & Tracking - Report

Executive Summary. This submission implements a complete computer vision pipeline that (i) detects the cricket ball centroid per frame when visible, (ii) writes a per-frame CSV annotation file in the required format, and (iii) produces an annotated MP4 with centroid + trajectory overlay. The approach is classical CV (HSV masking + contour filtering) paired with lightweight tracking and is fully reproducible via the provided repository scripts.

1 Objective & Deliverables

Goal: Build a complete computer vision system to detect and track a cricket ball in videos captured from a **single static camera**. The system produces:

- **Per-frame centroid detections** (x, y) whenever the ball is visible.
- **Annotation file** (CSV) with `frame, x, y, visible`.
- **Processed video** with the ball centroid and trajectory overlaid.
- **Reproducible code** and clear run instructions.

2 Data & Assumptions

Input: Cricket videos recorded from a fixed camera.
Assumptions:

- Camera is static throughout each clip (no pan/zoom).
- Ball appears as a small **bright object** (often white or red/orange) relative to background.
- Ball motion is smooth across frames; sudden jumps are unlikely.
- Ground-truth labels are not provided; evaluation is qualitative.

3 System Overview

Pipeline:

$$Video \rightarrow Frame - wiseDetection \rightarrow Tracking/Trajectory \rightarrow CSV + AnnotatedMP4$$

3.1 Repository Layout (Reproducible)

Path	Purpose
`code/`	Inference pipeline, detector, tracker, utilities
`annotations/`	Generated per-video CSV files
`results/`	Processed MP4 videos with overlays
`requirements.txt / environment.yml`	Dependencies
`README.md`	Setup + run instructions
`report.pdf`	This report

4 Detection Method

A classical CV detector is used to avoid training overhead and maximize reproducibility.

4.1 Preprocessing

- **Downscaling:** Each frame is resized (e.g., $0.5\times$) to improve speed for high-resolution videos.
- **Color space:** Convert BGR $\rightarrow$ HSV for robust thresholding.

4.2 HSV Masking (Ball Candidate Regions)

Two masks are combined:

- **White-ish mask:** low saturation, high value (captures bright/white ball).
- **Red/Orange mask:** hue ranges around red/orange (captures red ball).

The combined mask is smoothed with Gaussian blur to suppress noise and stabilize contours.

4.3 Contour Filtering & Selection

From the mask, connected components are extracted via contours and filtered by:

- **Area bounds** to remove small noise blobs and large non-ball regions.
- **Radius bounds** using the minimum enclosing circle to enforce plausible ball size.

If multiple candidates remain, the **largest valid contour** is selected as the ball.

4.4 Centroid Estimation

The centroid is taken as the center of the minimum enclosing circle of the selected contour, then mapped back to original resolution if downscaled.

5 Tracking & Trajectory

A lightweight tracker maintains:

- **Last known valid position** (for continuity).
- **Trajectory history** (only frames where a detection exists).

Overlay policy:

- Draw a **green circle** at the detected centroid when visible.
- Draw a **blue polyline** connecting historical detections to visualize trajectory.

6 Output Format & Fallback Logic

6.1 CSV Annotation Format

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

6.2 Visibility Flag

- `visible=1`: ball centroid detected; `x,y` are valid coordinates.
- `visible=0`: ball not detected; `x=-1, y=-1`.

6.3 Fallback Rules

1. **No detection in a frame:** output `-1,-1,0`; do not add to trajectory history.
2. **Multiple candidates:** select the best by validity constraints + maximal area.
3. **Temporary occlusion:** trajectory remains consistent because only confident detections are appended.

7 Evaluation (Qualitative)

Without ground-truth annotations, evaluation is based on:

- Visual correctness of centroid placement in frames where the ball is visible.
- Smoothness and plausibility of the overlaid trajectory.
- Behaviour under brief missed detections (proper `visible=0` marking).

8 Reproducibility

All hyperparameters are centralized and easy to tune (HSV thresholds, scale factor, area/radius bounds). The same videos + same environment reproduce identical outputs.

8.1 How to Run

```
pip install -r requirements.txt

cd code
python pipeline.py --input ../data/25_nov_2025
```

Outputs:

- `annotations/<video>.csv`
- `results/<video>_annotated.mp4`

9 Limitations & Future Work

- HSV thresholding can degrade under extreme illumination changes or similar-colored clutter.
- A **Kalman filter** can improve interpolation during occlusions.
- A trained **small-object detector** (e.g., YOLO) on a labeled subset would improve robustness.