Cricket Ball Detection and Trajectory Tracking

Dhana Lakshmi - AI22BTECH11012

February 18, 2026

1 Problem Statement

The objective of this project is to detect the centroid of a cricket ball in each frame of a video recorded from a single static camera. The system must:

- Detect the cricket ball centroid in each visible frame.
- Output a per-frame annotation file (CSV or JSON) containing:
  - frame_index
  - x_centroid
  - y_centroid
  - visibility_flag
- Generate a processed video with the ball trajectory overlaid.

2 Approach Overview

The solution uses a classical computer vision pipeline based on:

1. Frame extraction from input video.
2. Color-based segmentation of the cricket ball.
3. Contour filtering using geometric constraints.
4. Trajectory-based tracking using temporal consistency.
5. Annotation generation and output video creation.

This approach avoids heavy neural networks and relies on lightweight, real-time image processing techniques.

3 Pipeline Description

3.1 Frame Extraction

Each input video is read using OpenCV. Frames are extracted sequentially and stored in memory.

- OpenCV `VideoCapture` is used.
- Frames are stored in a list.
- FPS is recorded for output video generation.

3.2 Ball Detection

Ball detection is based on color segmentation in HSV space.

3.2.1 Color Segmentation

The cricket ball is detected using two color ranges:

- Red ball range
- White ball range

Binary masks are created using:

$$mask = mask_{red} \cup mask_{white}$$

Morphological operations (opening and closing) are applied to remove noise.

3.2.2 Contour Filtering

Contours extracted from the mask are filtered using:

- Area constraint:
$$40 < area < 500$$
- Circularity:
$$circularity = \frac{4\pi A}{P^2}$$
Only contours with circularity > 0.7 are kept.
- Aspect ratio constraint:
$$0.85 < \frac{width}{height} < 1.15$$

These constraints eliminate:

- Gloves
- Pads
- Field markings
- Other white or red objects

3.3 Spatial Constraints

To avoid false detections:

- Bottom 15% of the frame is ignored.
- Top 25% of the frame is ignored.

This removes:

- Players' gloves
- Field distractions

3.4 Temporal Tracking

A simple trajectory-based tracking method is used.

- The ball position from the previous frame is stored.
- New detections are accepted only if:

$$8 < distance < 70 \text{ pixels}$$

This ensures:

- Smooth trajectory
- Rejection of sudden false detections

3.5 Trajectory Visualization

- Detected centroids are stored.
- A polyline is drawn to represent the trajectory.

4 Annotation Output

For each frame, the following values are stored:

- frame_index
- x_centroid
- y_centroid
- visibility_flag

If the ball is not detected:

$$x = -1, \quad y = -1, \quad visibility = 0$$

Annotations are saved as a CSV file.

5 Multi-Video Processing

The system automatically processes all videos inside a directory.
For each input video:

- Frames are extracted.
- Ball detection is performed.
- Output video is generated.
- Annotation CSV is saved.

6 Implementation Details

6.1 Libraries Used

- OpenCV
- NumPy
- Pandas
- Python standard libraries

6.2 Video Format Handling (.mov to .mp4)

During experimentation, some input videos were provided in `.mov` format. In certain environments (e.g., Google Colab or limited OpenCV builds), `.mov` files may not open correctly due to codec compatibility issues.

To ensure reliable processing, `.mov` files were converted to `.mp4` format using FFmpeg before running the detection pipeline.

The conversion command used:

```
ffmpeg -i input.mov output.mp4
```

After conversion, the pipeline processes the `.mp4` files without any modification to the core detection algorithm.

6.3 Output Files

For each video:

- `video_name_trajectory.mp4`
- `video_name_annotations.csv`

7 Key Design Decisions

7.1 Color-Based Detection

Chosen because:

- Lightweight and fast.
- No training data required.
- Works well under consistent lighting.

7.2 Geometric Filtering

Circularity and aspect ratio filters were added to:

- Remove gloves and pads.
- Improve detection precision.

7.3 Trajectory-Based Validation

Temporal constraints ensure:

- Smooth ball motion.
- Rejection of sudden false detections.

8 Assumptions

- Camera is static.
- Ball color is either red or white.
- Ball size falls within a known pixel range.
- No severe lighting changes.

9 Limitations

- May fail under extreme lighting.
- Cannot handle heavy occlusion.
- Not robust to unusual ball colors.

10 Possible Improvements

- Add Kalman filter for smoother tracking.
- Train a deep learning object detector.
- Use optical flow for motion-based tracking.

11 Conclusion

A lightweight computer vision pipeline was developed to detect and track a cricket ball in video frames. The system produces per-frame centroid annotations and a processed video with trajectory overlay. The approach is efficient, reproducible, and satisfies the assignment requirements.