Cricket Ball Tracking

EdgeFleet Challenge Submission Report

Pradeep Chandra 25173

1. Executive Summary

This project implements a **vanilla cricket-ball tracker** using:

- YOLOv11s (custom-trained)
- Re-detect crop based on last known point
- A 4-state Kalman filter for smoothing and prediction

The tracker processes all videos inside `input_videos/` and generates:

- A processed video with green trail + red locked box
- A CSV file with per-frame ball coordinates (`frame, x, y, visible`)

Due to serious health issues, dataset preparation and experimentation time were limited to **one day**. As a result, the model is trained on only ≈ 300 frames, impacting recall and tracking stability. This report documents the implemented system, observed failure cases, and potential improvements.

2. System Overview

The full pipeline is shown conceptually in Fig. 1.

3. YOLO Training Details

A custom YOLOv11s model was trained on $\sim$300 manually annotated images (15 videos $\times$ 20 frames). Training used:

- Resolution: 1280px
- Augmentations: mosaic, mixup, copy-paste, geometric + color jitter
- 100 epochs

Dataset size was insufficient to generalize across lighting, ball size, occlusion, and motion blur, reducing recall in distant-ball frames.

```
1. Load video and detect ball color (red/white)
2. For each frame:
     • Preprocess depending on ball color
     • Run YOLO detector
     • If YOLO misses:
         – After $k$ frames, run re-detect crop
     • If still missing:
         – Fall back to Kalman prediction
     • Smooth red bounding box using interpolation
     • Log results to CSV and write video output
```

Figure 1: Overall tracking pipeline.

4. Inference Pipeline Overview

The full detection + tracking logic includes:

1) **Ball-color detection** using simple HSV heuristics.
2) **Preprocessing**:
    - CLAHE for white-ball videos
    - Contrast enhancement for red-ball videos
3) **YOLO detection** on each frame.
4) **Re-detect crop**:

$$\text{Crop} \rightarrow \text{Upscale} \rightarrow \text{YOLO}$$

5) **Kalman filter prediction** when YOLO fails.
6) **Locked red-box smoothing**:

$$B_t = B_{t-1} + \alpha(B_t^* - B_{t-1})$$

7) **CSV Output**.

5. Observed Failure Cases

5.1 Pad Occlusion (White Ball + White Pad)

When the white ball overlaps visually with the white pad, the detector fails completely. Example:

```
7,904,165,1
8,-1,-1,0
9,-1,-1,0
10,923,175,1
```

YOLO loses the ball; Kalman predicts incorrectly; re-detect crop searches in the wrong region.

5.2 Sudden Direction Changes

After pad or bat impact, ball deflection violates Kalman's constant-velocity assumption.

5.3 Sparse Detections

Videos such as Video-12 show 50+ consecutive frames of misses before the ball becomes big enough.

5.4 Lighting Variation

Overexposed or shadowed white balls appear indistinguishable from noise.

5.5 Frame-Edge Exits

The tracker continues predicting outside the visible area until its state diverges.

6. Root Cause Analysis

- **Tiny training dataset** (∼300 images)
- **No occlusion-handling logic**
- **Kalman filter too simple for cricket-ball dynamics**
- **Re-detect crop too small**
- **White ball lacks color contrast against pads/background**

7. Limitations of the Vanilla Tracker

- Cannot handle pad occlusions
- Cannot recover from abrupt deflections
- Struggles with tiny far-away ball detections
- Small-object recall too low due to limited YOLO training
- No physics model or multi-hypothesis tracker

8. Planned Improvements

1. Expand dataset (60–150 frames per video)
2. Train with stronger augmentations for white-ball scenarios
3. Incorporate occlusion logic or multi-hypothesis tracking
4. Add optical flow or motion-guided proposals
5. Train larger model (YOLO11m or YOLO11l)
6. Improve re-detection window and fallback strategies

9. Author's Note on Timeline Constraints

During the preparation of this work, I experienced an unexpected and serious health issue that substantially reduced the time I was able to dedicate to this project. Because of this, the submitted system reflects a **functional baseline** rather than the more sophisticated and complete tracker I originally intended to build.

Despite the limited time, I ensured that the full pipeline—YOLO-based detection, re-detection logic, Kalman-filter prediction, ball-color adaptation, and CSV generation—was implemented and tested across all provided videos. However, several planned enhancements remain incomplete, including robust occlusion handling, improved direction-change recovery, expanded dataset preparation, and physics-assisted trajectory modeling.

It is important to emphasize that the current limitations arise not from conceptual challenges, but from the compressed development window caused by my health condition. Under normal circumstances, the complete system would have been delivered.

To address this responsibly, I will finalize the remaining enhancements within the next **two days**. This includes:

- Increasing dataset size and retraining YOLOv11s for higher recall
- Implementing pad-occlusion recovery mechanisms
- Improving fallback tracking for white-ball frames
- Enhancing temporal smoothing and stabilization
- Updating this report with improved results and error analysis

This submission should therefore be viewed as an honest and transparent baseline version, with clear documentation of its issues and a concrete plan for completing the full system promptly.

10. Conclusion

A complete baseline tracking system using YOLOv11s, re-detection, and Kalman filtering was implemented. Although functional, its performance is limited by training-data size and lack of occlusion-handling mechanisms. The report provides a full breakdown of failures and future improvements. A refined version of the system will be completed soon as outlined.