Cricket Ball Detection and Tracking in Broadcast Videos

A Robust End-to-End Detection and Motion-Gated Tracking System

Abstract

Tracking a cricket ball in broadcast videos is a challenging computer vision problem due to the ball's extremely small size, high velocity, severe motion blur during delivery, frequent occlusions, and visual similarity to background objects such as helmets and caps. This project presents a complete end-to-end pipeline for cricket ball detection and tracking that integrates a YOLOv8-based object detector with a motion-gated Kalman filter. The system is intentionally conservative: it produces trajectories and annotations only when sufficient visual evidence exists and avoids hallucinated tracking. This report consolidates the complete project description, workflow, exploratory data analysis (EDA), encountered challenges, implemented solutions, limitations, and future improvement directions.

1. Problem Statement and Objectives

The goal of this project is to design a system that can:

- Detect a cricket ball in broadcast-style cricket videos
- Track the ball across frames when visually feasible
- Generate:
  - Frame-wise CSV annotations: \((frame, x, y, visible)\)
  - Output videos with ball centroid and trajectory overlays
- Handle real-world challenges such as:
  - Motion blur during bowling
  - Very small object size
  - Occlusions and camera motion
  - False positives caused by helmets, caps, and background clutter

A key design principle of this system is:

> ***It is better to output no tracking than incorrect or hallucinated tracking.***

2. Repository and System Structure

The final project structure is organized as follows:

```
project_root/
│
├── code/
│   ├── main.py           # Entry point (batch inference)
│   ├── inference.py      # YOLOv8 inference wrapper
│   ├── kalman.py         # Constant-velocity Kalman filter
│   ├── tracker.py        # Motion-gated tracking logic
│   ├── utils.py          # Visualization utilities
│
├── test_videos/          # Input videos for inference
│
├── annotations/          # Output CSV annotation files
│   ├── 1.csv
│   ├── 2.csv
│   └── ...
│
├── results/              # Output videos with overlays
│   ├── 1_tracked.mp4
│   ├── 2_tracked.mp4
│   └── ...
│
├── best.pt               # Trained YOLOv8 model
├── requirements.txt      # Python dependencies
└── report.md             # Project documentation
```

---

3. End-to-End System Workflow

The complete pipeline consists of the following stages:

1. Training data analysis and preparation
2. Training a YOLOv8-based cricket ball detector
3. Exploratory data analysis on inference videos
4. Frame-wise ball detection during inference
5. Detection post-processing and gating
6. Kalman filter-based temporal tracking
7. Trajectory visualization and CSV annotation generation
8. Batch inference over multiple videos

Each stage is designed to mitigate specific real-world challenges.

---

4. Training Dataset and Exploratory Data Analysis (EDA)

4.1 Dataset Description (Roboflow)

The detection model was trained using a Roboflow cricket ball detection dataset:

- Training images: 199
- Validation images: 28
- Test images: 0

The dataset follows the YOLO annotation format.

4.2 Image Properties

- Resolution: 640 × 640
- Aspect ratio: 1:1
- Channels: RGB (3 channels)

All images are resized to a fixed resolution, which simplifies training but introduces a scale mismatch at inference time.

4.3 Bounding Box Statistics

Bounding boxes are normalized to image dimensions. Observed statistics:

Width:

- Minimum ≈ 0.016
- Mean ≈ 0.046

Height:

- Minimum ≈ 0.021
- Mean ≈ 0.052

This confirms that the ball occupies a very small fraction of the image.

4.4 Key Observations from Training EDA

- Dataset consists of still images only
- No temporal or motion information
- Limited representation of motion-blurred delivery frames

Implication: The detector learns appearance but not motion, making temporal tracking essential during inference.

---

5. Test Video Dataset and Inference EDA

5.1 Video Metadata

The inference dataset contains multiple broadcast cricket videos with:

- Resolutions ranging from 1920×1080 to 2560×1600
- Frame rates between 25 FPS and ~57 FPS
- Predominantly 16:9 aspect ratios

This represents a significant domain shift from the training data.

5.2 Empirical Inference Analysis

Using the trained detector:

- Detection rate ≈ 4% of frames
- Mean detected ball size ≈ 70–85 pixels
- Mean pixel velocity ≈ 13 px/frame
- Maximum velocity spikes ≈ 60–70 px/frame

5.3 Key Inference Observations

- Reliable detections mainly occur after the ball is hit
- Delivery-phase detections are sparse due to motion blur
- Frequent false positives occur on helmets, caps, and static circular regions

---

6. Ball Detection Module

6.1 Model Selection

YOLOv8-Nano, pretrained and fine-tuned on the Roboflow dataset

6.2 Rationale

- Lightweight and efficient
- Good performance on small objects
- Suitable for CPU or limited-GPU environments

6.3 Limitations of Detection Alone

Detection-only approaches were found to:

- Miss the ball in most delivery-phase frames
- Produce isolated detections without temporal consistency
- Be highly susceptible to false positives

These limitations motivated the use of a tracking-based pipeline.

---

7. Kalman Filter-Based Tracking

7.1 Motivation

Kalman filtering is used to:

- Smooth noisy detections
- Handle missing measurements
- Enforce temporal consistency

7.2 State Representation

A constant-velocity model is used:

$x_t = [x_t, y_t, v_{x,t}, v_{y,t}]^T$

The model assumes linear motion over short time intervals.

---

8. Motion-Gated Tracking Pipeline

To avoid hallucinated tracking, multiple gating mechanisms are applied.

8.1 Spatial Gating

Detections are rejected if:

$y < 0.15 \times H$

This removes false positives near the top of the frame.

8.2 Motion Gating

After initialization, detections must satisfy:

$\|\Delta p\| \geq \tau$

This removes static or near-static false positives.

8.3 Evidence-Based Initialization

The tracker is initialized only after:

- A minimum number of consecutive valid detections
- A minimum total detection count

This prevents tracker lock-on to accidental detections.

8.4 Limited Prediction Horizon

Kalman predictions are limited to a very small number of frames (typically one) to avoid long straight-line extrapolations.

---

9. Annotation and Visualization

9.1 CSV Annotation Format

Each video produces a CSV file with structure:

```
frame,x,y,visible
```

Where:

- `visible = 1`: detector measurement used
- `visible = 0`: predicted or ball not visible
- `x = y = -1`: tracking disabled

This explicitly encodes uncertainty.

9.2 Video Output

- Red dot: ball centroid
- Blue polyline: tracked trajectory
- No overlay when tracking is disabled

---

10. Problems Encountered and Solutions

Problem	Solution
False Positives on Helmets and Caps	Spatial gating, motion gating, and evidence-based initialization
Sparse and Noisy Detections	Kalman filtering with short prediction horizon

Problem	Solution
Hallucinated Straight-Line Trajectories	Aggressive tracker reset and limited prediction frames
Domain Shift Between Training and Inference	Conservative tracking policy and explicit visibility modeling

---

11. Limitations

- Delivery-phase tracking remains unreliable due to motion blur
- Performance depends heavily on detector quality
- Single-camera setup limits physical modeling
- Some videos produce no tracking output (expected behavior)

---

12. Future Improvements

Potential enhancements include:

\begin \item Augmenting training data with labeled video frames \item Adding hard negative samples (helmets, caps) \item Multi-scale training for small-object robustness \item Physics-aware motion models \item Optical-flow-assisted detection \item Sequence-based deep learning models \end

---

13. Conclusion

This project demonstrates a realistic and principled approach to cricket ball tracking in broadcast videos. By combining deep-learning-based detection with motion-gated Kalman tracking and conservative decision-making, the system avoids hallucinated outputs and produces interpretable results. The pipeline reflects real-world sports analytics constraints and provides a strong foundation for future improvements.

The key achievement is not just detecting and tracking the ball, but doing so with deliberate conservatism—recognizing when the evidence is insufficient and explicitly declining to produce unreliable annotations. This principled approach is essential for building trust in computer vision systems deployed in production sports analytics workflows.

References

[1] Ultralytics. YOLOv8: A State-of-the-Art Real-Time Object Detection Framework. https://github.com/ultralytics/ultralytics (https://github.com/ultralytics/ultralytics)

[2] Roboflow. Cricket Ball Detection Dataset. https://roboflow.com (https://roboflow.com)

[3] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1), 35–45.

[4] Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *arXiv preprint arXiv:1804.02767*.

[5] Bewley, A., Ge, Z., Ott, L., Rameshan, F., & Reid, I. (2016). Simple online and realtime tracking. *ICIP*.