Cricket Ball Detection & Trajectory Tracking System

EdgeFleet.AI – AI/ML Assessment

Submitted By

Name: Omkar Lahore

Roll Number: 205324015

Course: M.Sc. Computer Science

Department: Computer Applications

Institution: National Institute of Technology, Tiruchirappalli

Submission Details

Date of Submission: 26/12/2025

Submitted To

EdgeFleet.AI

Table of Contents

1. Introduction

Tracking a cricket ball from video is a challenging computer vision task due to the ball's **small size**, **high speed**, and frequent **motion blur**. Traditional tracking approaches often fail when detections are missed or when the ball is temporarily occluded.

This project presents a **robust end-to-end computer vision system** that combines **deep learning–based object detection** with **classical state estimation techniques** to reliably detect and track a cricket ball from videos captured using a **single, fixed camera**.

The work was carried out as part of the **EdgeFleet.AI AI/ML Assessment**, with emphasis on:

- Robust detection
- Smooth trajectory estimation
- Clean output generation
- Full reproducibility

2. Problem Definition

The objective of this project is to design and implement a system that:

- Detects a cricket ball in each frame of a video
- Extracts the ball centroid (x, y)
- Tracks the ball across frames even when detections are missing
- Generates:
    - A processed video with trajectory overlay
    - A structured per-frame annotation file (CSV)

The system must work under real-world constraints such as motion blur, fast ball movement, and temporary occlusion.

3. Dataset Collection and Preparation

3.1 Training Dataset

The object detection model was trained using a **custom image dataset** created and managed through **Roboflow**.

Approximately **10,000 images** of cricket balls were:

- Collected from multiple sources
- Manually annotated with bounding boxes
- Curated to ensure annotation quality

The dataset was designed to cover **maximum variability**, including:

- Different camera angles
- Varying ball sizes and scales
- Diverse lighting conditions
- Motion blur scenarios
- Complex and cluttered backgrounds

This diversity was crucial for training a detector capable of generalizing to unseen match videos.

3.2 Data Preparation

The dataset was split into:

- Training set
- Validation set

Standard data augmentation techniques provided by Roboflow (scaling, flipping, brightness variation) were used to improve generalization and robustness.

3.3 Testing Dataset (Videos)

The trained model was evaluated using **unseen cricket videos** captured from a **single static camera**.

Important distinction:

- **Images** → used only for training
- **Videos** → used only for testing and evaluation

This separation avoids data leakage and provides a realistic evaluation of real-world performance.

4. System Architecture Overview

The proposed system is designed using a modular and two-stage architecture, where model training and video-based inference are handled as independent yet connected components. This separation improves reproducibility, simplifies debugging, and allows each stage to be optimized independently.

Stage 1: Training

The training stage focuses on learning a robust visual representation of the cricket ball using a labeled image dataset.

- Input:
  A large-scale, manually annotated image dataset curated using Roboflow, consisting of approximately 10,000 labeled images.
  The dataset includes diverse variations in:
    - Camera angles
    - Ball sizes and scales
    - Lighting conditions
    - Motion blur and background complexity
- Model:
  A YOLOv8-based object detection model is trained to detect the cricket ball as a single class.
  YOLO’s single-stage detection architecture enables efficient learning of spatial features required for small and fast-moving object detection.
- Output:
  The output of this stage is a trained model file (best.pt), which encapsulates the learned weights and can be reused for inference on unseen data.

This stage is executed independently and does not rely on any video data, ensuring a clean separation between training and evaluation.

Stage 2: Inference and Tracking

The inference stage applies the trained detection model to unseen cricket videos and estimates the ball trajectory over time.

- Input:
  Cricket videos recorded using a single fixed camera, which were not used during training.
- Detection:
  Each video frame is passed through the trained YOLO model to obtain bounding box detections and confidence scores.
  Only detections belonging to the cricket ball class are considered, and the highest-confidence detection is selected per frame.
- Tracking:
  A Kalman Filter is employed to track the ball centroid across frames.
  This helps:
  - Smooth noisy detections
  - Predict ball position when detections are temporarily missing
  - Maintain a continuous trajectory
- Output:
  - A processed video with:
    - Ball bounding box
    - Centroid visualization
    - Trajectory overlay
  - A per-frame CSV annotation file containing:
    - Frame index
    - Ball centroid coordinates
    - Visibility flag

5. Model Selection and Training

5.1 Choice of YOLO

YOLO (You Only Look Once) was selected due to:

- Single-stage detection (fast inference)
- Strong performance on small objects
- Easy integration with video pipelines

Since the task focuses only on cricket ball detection, the model was trained for a **single class**, reducing classification ambiguity.

5.2 Training Strategy

The model was trained using the Roboflow-exported dataset with:

- High-resolution images
- Careful hyperparameter tuning
- Validation-based early stopping

The final trained model (best.pt) demonstrated strong localization performance on validation data.

5.3 Dataset–Model Alignment

Because cricket balls occupy very few pixels in many frames, the training dataset emphasized:

- Scale variation
- Motion blur representation
- Diverse backgrounds

6. Ball Detection Pipeline

The ball detection pipeline is responsible for identifying the cricket ball in each frame of the input video. Since the cricket ball is a small, fast-moving object, special care is taken to ensure reliable detection while minimizing false positives.

For each frame in the input video, the following steps are executed:

6.1 Frame Processing

Each video is processed frame by frame to ensure fine-grained temporal analysis. Every frame is treated as an independent image and passed to the trained object detection model. This approach allows the system to capture rapid changes in ball position caused by high-speed motion.

6.2 YOLO-Based Detection

The processed frame is forwarded to the trained YOLO model, which performs object detection in a single forward pass.
The model outputs:

- Bounding box coordinates
- Confidence scores
- Class predictions

YOLO's single-stage architecture enables efficient and accurate detection, making it suitable for high-resolution video frames.

6.3 Class Filtering

Since the model is trained specifically for cricket ball detection, only detections belonging to the cricket ball class are considered during inference.
All other detections (if any) are discarded.

This filtering step significantly reduces noise and prevents unrelated objects from affecting the tracking pipeline.

6.4 Best Detection Selection

In cases where multiple cricket ball detections are produced in a single frame, the detection with the highest confidence score is selected as the final detection for that frame.

This strategy:

- Improves detection reliability
- Reduces ambiguity when multiple bounding boxes overlap
- Ensures a single, consistent centroid is passed to the tracking stage

6.5 Output of Detection Stage

The final output of the detection pipeline for each frame consists of:

- A single bounding box (if detection is available)
- The associated confidence score
- A flag indicating whether the ball was detected

These outputs serve as inputs to the tracking and trajectory estimation module, which further refines the ball's motion across frames.

7. Tracking and Trajectory Estimation

7.1 Kalman Filter Design

- State vector: (x, y, vx, vy)
- Measurement: (x, y)
- Assumes constant velocity motion

7.2 Tracking Logic

- If detection is available:
    - Kalman filter is corrected using measured centroid
- If detection is missing:
    - Kalman prediction is used

This produces a smooth and continuous trajectory across frames.

8. Fallback Logic and Error Handling

To prevent unrealistic tracking behavior:

- A motion gating threshold limits sudden jumps
- Detections far from the previous position are rejected
- Predictions are only trusted for short durations

This logic increases robustness in noisy or ambiguous frames.

9. Output Generation

9.1 Processed Video

The output video includes:

- Bounding box around detected ball
- Centroid visualization
- Continuous trajectory overlay

9.2 CSV Annotation File

Each frame produces one CSV entry:

frame,x,y,visible

- visible = 1 → detection used
- visible = 0 → prediction used

This structured format enables further analysis and evaluation.

10. Experimental Evaluation

The trained model and tracking pipeline were evaluated on multiple unseen cricket videos.

Observations

- High detection accuracy when ball is visible
- Smooth trajectory even during short detection gaps
- Stable performance across different videos

The system successfully met all functional requirements of the assessment.

11. Challenges Faced and Mitigation Strategies

Challenges

- Very small object size
- Fast motion and blur
- Temporary occlusions

Mitigation

- Large and diverse training dataset
- Confidence-based detection selection
- Kalman filter smoothing
- Motion gating logic

12. Limitations

- Assumes single ball
- Designed for static camera
- No 3D trajectory estimation
- No quantitative metric computation included

13. Future Scope and Improvements

- Physics-aware trajectory modeling
- Multi-camera 3D reconstruction
- Optical flow–based refinement
- Automated quantitative evaluation
- Real-time deployment optimization

14. Conclusion

This project presents an end-to-end approach for cricket ball detection and trajectory tracking by integrating a deep learning–based object detection model with classical state estimation techniques. By training a YOLO-based detector on a large, custom-labeled image dataset and applying it to unseen video data, the system demonstrates strong generalization to real-world match scenarios.

The incorporation of a Kalman Filter plays a crucial role in maintaining trajectory continuity, particularly in challenging situations involving motion blur, partial occlusion, or temporary detection failures. This hybrid approach effectively balances detection accuracy with temporal consistency, resulting in smooth and reliable ball trajectories across frames.

A key strength of the system lies in its modular design, where training, detection, and tracking are clearly separated. This not only improves reproducibility and interpretability but also allows individual components to be enhanced or replaced without affecting the overall pipeline. The structured outputs in both visual and tabular formats further enable downstream analysis and future extensions.

Overall, the project demonstrates how combining data-driven learning with principled tracking methods can yield robust performance in complex video-based tracking tasks. The system provides a solid foundation for future work in sports analytics, trajectory modeling, and real-time vision-based decision support.